

Федеральное государственное бюджетное образовательное учреждение  
высшего образования  
«КАЛИНИНГРАДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Е. Ю. Заболотнова  
С. А. Калинина

## **ПРОГРАММИРОВАНИЕ**

Учебно-методическое пособие по выполнению лабораторных работ  
для студентов очной формы обучения направления подготовки  
09.03.01 Информатика и вычислительная техника  
и очной и заочной форм обучения направления подготовки  
09.03.03 Прикладная информатика

Калининград  
Издательство ФГБОУ ВО «КГТУ»  
2022

УДК 004.9(075)

Рецензент:

кандидат экономических наук, и.о. зав. кафедрой прикладной информатики  
ФГБОУ ВО «Калининградский государственный технический университет»  
М. В. Соловей

Заболотнова, Е. Ю.

Программирование: учебно-методическое пособие по выполнению лабораторных работ для студентов очной формы обучения направления подготовки 09.03.01 Информатика и вычислительная техника и очной и заочной форм обучения направления подготовки 09.03.03 Прикладная информатика / Е. Ю. Заболотнова, С. А. Калинина.– Калининград: Изд-во ФГБОУ ВО «КГТУ», 2022. – 90 с.

Данное учебно-методическое пособие содержит лабораторные работы по дисциплине: задания, методические указания по выполнению работ, структуру отчета и требования к его оформлению, приведены контрольные вопросы и порядок защиты лабораторных работ

Учебно-методическое пособие рассмотрено и одобрено в качестве локального электронного методического материала кафедрой прикладной информатики института цифровых технологий ФГБОУ ВО «Калининградский государственный технический университет» 19 сентября 2022 г., протокол № 3

Учебно-методическое пособие рекомендовано к использованию в качестве локального электронного методического материала в учебном процессе методической комиссией ИЦТ 20 сентября 2022 г., протокол № 6

УДК.004.9(075)

© Федеральное государственное  
бюджетное образовательное  
учреждение высшего образования  
«Калининградский государственный  
технический университет», 2022 г.  
© Заболотнова Е. Ю., Калинина С. А.,  
2022 г.

**ОГЛАВЛЕНИЕ**

1.	Введение .....	6
2.	Требования к отчету и порядок защиты лабораторных работ.....	6
3.	Лабораторная работа №1. Линейные алгоритмы и программы .....	7
	Общие сведения.....	7
	Теоретическое введение .....	7
	Задание к лабораторной работе .....	10
	Методические указания и порядок выполнения работы .....	10
	Индивидуальное задание .....	11
4.	Лабораторная работа №2. Использование структур выбора .....	11
	Общие сведения.....	11
	Теоретическое введение .....	12
	Задание к лабораторной работе .....	13
	Методические указания и порядок выполнения работы .....	13
	Индивидуальное задание .....	14
5.	Лабораторная работа №3. Использование структур повторения. Циклы. ....	14
	Общие сведения.....	14
	Теоретическое введение .....	15
	Задание к лабораторной работе .....	16
	Методические указания и порядок выполнения работы .....	16
	Индивидуальное задание .....	19
6.	Лабораторная работа №4. Списки. Операции, методы и функции Списков ..	19
	Общие сведения.....	19
	Теоретическое введение .....	19
	Задание к лабораторной работе .....	22
	Методические указания и порядок выполнения работы .....	22
	Индивидуальное задание .....	23
7.	Лабораторная работа №5. Работа с матрицами.....	24
	Общие сведения.....	24
	Теоретическое введение .....	24
	Задание к лабораторной работе .....	26
	Методические указания и порядок выполнения работы .....	26
	Индивидуальное задание .....	27
8.	Лабораторная работа №6. Строки. Операции, методы и функции строк.....	27
	Общие сведения.....	27
	Теоретическое введение .....	28
	Задание к лабораторной работе .....	30
	Методические указания и порядок выполнения работы .....	30

Индивидуальное задание .....	31
9. Лабораторная работа №7. Функции. Аргументы функции. Вызов функции. Локальные и глобальные переменные .....	32
Общие сведения.....	32
Теоретическое введение .....	32
Задание к лабораторной работе .....	34
Методические указания и порядок выполнения работы .....	34
Индивидуальное задание .....	36
10. Лабораторная работа №8. Словари. Обработка исключений .....	36
Общие сведения.....	36
Теоретическое введение .....	37
Задание к лабораторной работе .....	40
Методические указания и порядок выполнения работы .....	40
Индивидуальное задание .....	42
11. Лабораторная работа №9. Основные методы для работы с файлами .....	43
Общие сведения.....	43
Теоретическое введение .....	43
Задание к лабораторной работе .....	49
Методические указания и порядок выполнения работы .....	49
Индивидуальное задание .....	51
12. Лабораторная работа №10. Графика в Python .....	52
Общие сведения.....	52
Теоретическое введение .....	52
Задание к лабораторной работе .....	55
Методические указания и порядок выполнения работы .....	56
Индивидуальное задание .....	58
13. Лабораторная работа №11. Анимация в Python .....	59
Общие сведения.....	59
Теоретическое введение .....	59
Задание к лабораторной работе .....	60
Методические указания и порядок выполнения работы .....	60
Индивидуальное задание .....	62
14. Лабораторная работа №12. Введение в объектно-ориентированное программирование. Создание классов. Атрибуты и методы.....	63
Общие сведения.....	63
Теоретическое введение .....	63
Задание к лабораторной работе .....	66
Методические указания и порядок выполнения работы .....	66
Индивидуальное задание .....	67

15. ЛАБОРАТОРНАЯ РАБОТА №13. Принципы объектно–ориентированного программирования. Наследование. Инкапсуляция.....	68
Общие сведения.....	68
Теоретическое введение .....	69
Задание к лабораторной работе .....	72
Методические указания и порядок выполнения работы.....	72
Индивидуальное задание.....	74
16. ЛАБОРАТОРНАЯ РАБОТА №14. Создание графического интерфейса пользователя (GUI) средствами библиотеки tkinter .....	75
Общие сведения.....	75
Теоретическое введение .....	76
Задание к лабораторной работе .....	81
Методические указания и порядок выполнения работы.....	82
Индивидуальное задание.....	85
17. Литература.....	88
18. Заключение .....	89

## 1. ВВЕДЕНИЕ

Данное учебно-методическое пособие предназначено для студентов очной формы обучения направления 09.03.01 Информатика и вычислительная техника и студентов очной и заочной форм обучения направления 09.03.03 Прикладная информатика, изучающих дисциплину «Программирование».

**Цель** лабораторного практикума по дисциплине: знакомство студентов с практикой применения основных структур данных языка Python, создание графических примитивов и основ анимации изображений, изучение принципов объектно-ориентированного программирования, приобретение навыков построения графических пользовательских интерфейсов и событийного программирования на их основе.

Лабораторный практикум содержит 14 лабораторных работ.

Лабораторные работы проводятся в лабораториях, оснащенных персональными компьютерами с установленным языком программирования Python версии 3.6 или выше.

В результате выполнения лабораторных работ ожидается, что студенты научатся разрабатывать алгоритмы самостоятельно и будут способны писать и отлаживать программы разной категории сложности.

Для студентов заочной формы обучения направления 09.03.03 Прикладная информатика рекомендовано выполнение 1-3, 4 и 6 лабораторных работ.

## 2. ТРЕБОВАНИЯ К ОТЧЕТУ И ПОРЯДОК ЗАЩИТЫ ЛАБОРАТОРНЫХ РАБОТ

Отчет по лабораторной работе представляется в электронном или печатном виде и должен содержать:

- Название дисциплины, номер работы, тема работы.
- Фамилию студента с указанием группы, Дату выполнения.
- Фамилию преподавателя, дата защиты.
- Контрольные вопросы по теме.
- Индивидуальное задание.
- Текст программы.
- Скриншоты возможных результатов.

Для отметки преподавателя следует распечатать первую страницу.

Отчет предоставляется в электронном виде на занятии, первая страница с подписью преподавателя выставляется в ЭИОС.

Защита лабораторной работы состоит в проверке работоспособности программы в соответствии с заданием и ответов на контрольные вопросы по теме работы.

### 3. ЛАБОРАТОРНАЯ РАБОТА № 1. ЛИНЕЙНЫЕ АЛГОРИТМЫ И ПРОГРАММЫ

#### Общие сведения

##### *Цель:*

Освоить понятия: алгоритм, свойства алгоритма, способы записи алгоритма. Изучить правила ввода данных, вывода результатов и вычислений в Python.

##### *Материалы, оборудование, программное обеспечение:*

Компьютерные классы с установленным высокоуровневым языком программирования Python.

##### *Условия допуска к выполнению:*

Изучить конспект по теоретической подготовке, теоретическое введение работы.

##### *Критерии положительной оценки:*

Для успешной сдачи лабораторной работы следует показать и объяснить, схему алгоритма, программу, представить отчет в соответствии с требованиями и пройти защиту.

##### *Планируемое время выполнения:*

Аудиторное время выполнения (под руководством преподавателя): 2 ч.

Время самостоятельной подготовки: 1 ч.

#### Теоретическое введение

Как правило, для простой задачи схема алгоритма (и программа на языке Python), реализующая линейный вычислительный процесс, должна состоять из следующих частей:

- Ввод данных.
- Выполнение вычислений.
- Вывод результата.

Схема линейного алгоритма состоит из блоков начало – конец, ввода – вывода, вычисления.

Для создания переменной в Python достаточно присвоить некоторому идентификатору значение при помощи оператора присваивания « $\leftarrow$ ».

```
a = 10
```

```
b = 3.1415926
```

```
c = "Hello"
```

В этом примере используются переменные:

- переменная *a* хранит значение типа `int` (целое число),
- переменная *b* — типа `float` (действительное число),
- переменная *c* — типа `str` (строка),

Для считывания строки со стандартного ввода используется функция `input()`, которая считывает строку с клавиатуры и возвращает значение считанной строки, которое сразу же можно присвоить переменной:

```
a = input()
```

Функция `input` возвращает текстовую строку. Если нужно сделать так, чтобы переменные имели целочисленные значения, то сразу же после считывания выполняется преобразование типа при помощи функции `int()`:

```
a = int(a)
```

Можно объединить считывание строк и преобразование типа и добавить пояснительный текст:

```
a = int(input("Введите значение a: "))
```

Если несколько значений вводятся в строке, нужно применить метод `split()`, который разделяет строку на части по одному или двум пробелам.

При вводе чисел через пробел, можно указать тип:

```
a, b, c = map(int, input("a? b? c? ").split())
```

Для вывода данных используется функция `print()`. Выводит значения переменных, значения любых выражений. Выводимые значения разделяются одним пробелом.

Можно разделять выводимые значения двумя пробелами, любым другим символом, любой другой строкой, выводить их в отдельных строках или не разделять.

Для этого функции print передают специальный параметр, называемый sep. По умолчанию параметр sep равен строке из одного пробела.

Параметр sep указывается в списке вывода:

```
print (a, b, c, sep = ' __:__ ')
```

Для того, чтобы значения выводились с новой строки, нужно:

```
print (a, sep = '\n', b)
```

Для того чтобы значения переменных разных предложений print выводились в одной строке, нужен параметр end = ":

```
print (a, b, end = ")
```

```
print (c, d)
```

Для ограничения при выводе количества дробных знаков (тип float) рекомендуется указывать формат. Сравните варианты вывода:

```
k=1/7
```

```
print (k)
```

```
print ("%7.2f"% k)
```

Результат:

```
0.14285714285714285
```

```
0.14
```

```
>>>
```

При описании формата указано:

7 – количество позиций для числа (необязательный параметр),

2 – количество дробных знаков после точки,

f – тип числа (число – вещественное)

Последовательные действия описываются последовательными строками программы.

Первая строка в сценарии Python не должна иметь отступа.

В таблице представлены арифметические операторы, операторы присваивания и функции, которые могут потребоваться при выполнении задания:

Арифметические операторы	Описание	Примеры
+, -, *, /	Сложение, Вычитание, Умножение, Деление	
%	Деление по модулю (возвращает остаток)	6 % 2 в результате будет 0 13.2 % 5 в результате 3.2
**	Возведение в степень	-3 ** 2 в результате будет -9
//	Целочисленное деление	4 // 3 в результате будет 1

Операторы присваивания	Описание	Примеры
=		c = 23
+=	Прибавит значение правого операнда к левому и присвоит эту сумму левому операнду	c += a или: c = c + a
-= *= /= %= **= // =	Аналогично	c *= a или: c = c * a



Для использования математических, тригонометрических и логарифмических операций в Python предназначен встроенный модуль `math`. Для подключения модуля укажите:

```
import math
```

Некоторые из основных функций модуля:

`pow (num, power)`: возведение числа `num` в степень `power`

`sqrt (num)`: квадратный корень числа `num`

`ceil (num)`: округление числа до ближайшего наибольшего целого

`floor (num)`: округление числа до ближайшего наименьшего целого

`factorial (num)`: факториал числа

`degrees (rad)`: перевод радиан в градусы

`radians (grad)`: перевод градусов в радианы

`cos (rad)`, `sin(rad)`, `tan (rad)`: косинус, синус, тангенс угла в радианах

`acos (rad)`, `asin(rad)`, `atan (rad)`: арккосинус, арксинус, арктангенс угла в радианах

`log (n, base)`: логарифм числа `n` по основанию `base`

`log10 (n)`: десятичный логарифм числа `n`

Например:

```
import math
```

```
a= math.sqrt(256)
```

```
print ("Квадратный корень числа 256 = ", a)
```

В программе рекомендуется использовать комментарии.

Для одной строки (#):

```
#Работу выполнил студент 21-ИЭ-1 Алексеев А.
```

Или для нескольких строк (группа трех апострофов ''' '''):

```
'''
```

```
Программу составил студент 21-ВТ
```

```
Васильев В.
```

```
'''
```

*Литература:*

[1] - глава 2. Стр. 16-20

глава 3. Стр.20-27

глава 4. Стр. 26-36

*Контрольные вопросы для самопроверки:*

1. Зачем нужен тип переменной?
2. Почему желательно выводить на экран подсказку перед вводом данных?
3. Какие простые типы данных вы знаете?
4. Что такое приоритет операций? Зачем он нужен?
5. В каком порядке выполняются операции, если они имеют одинаковый приоритет?
6. Зачем используются скобки?
7. Чем отличаются операции `/`, `//` и `%` ?

## Задание к лабораторной работе

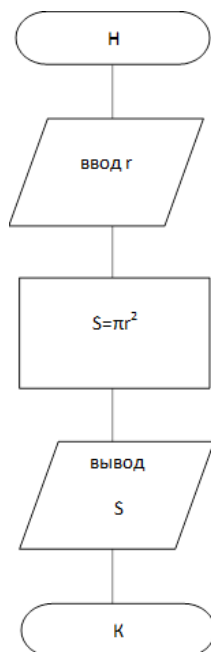
Согласно своему варианту составить для задачи линейный алгоритм, схему алгоритма, написать линейную программу на языке Python.

Оформить ввод значений и вывод результатов. Выполнить задачу для разных наборов данных.

## Методические указания и порядок выполнения работы

Например, вычислить площадь круга по заданному радиусу:  $s = \pi r^2$

Схема алгоритма может быть такой:



Примеры вариантов программ для вычисления площади круга по заданному радиусу:

```
# Николаев А. 21-ИЭ-1
print ("Радиус: ")
r = float (input())
p= 3.14 * r**2
print ("Площадь круга: ", p)
```

```
# Николаев А. 21-ИЭ-1
r = float (input("Радиус: "))
print ("Площадь круга: ", (3.14 * r**2))
```

```
# Николаев А. 21-ИЭ-1
import math
r = float (input("Радиус: "))
area = math.pi * math.pow(r, 2)
print("площадь круга: ", area)
```

## Индивидуальное задание

Номер варианта	Задача
0.	Напишите программу, которая вычисляет расстояние от начала координат до заданной точки
1.	Напишите программу, которая находит сумму, произведение и среднее арифметическое трех целых чисел, введенных с клавиатуры
2.	Напишите программу, которая по основанию и высоте равнобедренного треугольника вычисляет площадь и периметр
3.	Напишите программу, которая для числа $abcd$ находит сумму чисел $ab$ и $cd$ , число вводится с клавиатуры
4.	Напишите программу, которая вводит трехзначное число и разбивает его на цифры и находит их произведение. Например, при вводе числа 123 программа должна вывести 6
5.	Напишите программу, которая по двум введенным пользователем катетам вычислить длину гипотенузы
6.	Напишите программу, которая вводит трехзначное число и разбивает его на цифры и находит сумму квадратов этих чисел
7.	Напишите программу, которая по катету и гипотенузе вычисляет площадь и периметр прямоугольного треугольника
8.	Напишите программу, которая вычисляет площадь и периметр треугольника по данным трем сторонам, введенным с клавиатуры
9.	Напишите программу, которая находит среднее арифметическое трех вещественных чисел, заданных с клавиатуры
10.	Напишите программу, которая по заданным координатам двух точек на плоскости вычисляет расстояние между ними
11.	Напишите программу, которая по заданным диаметрам двух окружностей с общим центром вычисляет площадь образуемого ими кольца
12.	Напишите программу, которая по заданной стороне квадрата вычисляет его периметр и площадь
13.	Напишите программу, которая по заданным координатам центра и некоторой точки окружности вычисляет длину окружности и площадь круга, образованного ею
14.	Напишите программу, которая для числа $abcd$ находит сумму цифр, число вводится с клавиатуры

## 4. ЛАБОРАТОРНАЯ РАБОТА №2. ИСПОЛЬЗОВАНИЕ СТРУКТУР ВЫБОРА

### Общие сведения

#### *Цель:*

Получить навыки работы с условной инструкцией - основному инструменту выбора в Python.

#### *Материалы, оборудование, программное обеспечение:*

Компьютерные классы с установленным высокоуровневым языком программирования Python.

#### *Условия допуска к выполнению:*

Изучить конспект по теоретической подготовке, теоретическое введение работы.

*Критерии положительной оценки:*

Для успешной сдачи лабораторной работы следует показать и объяснить схему алгоритма, программу, представить отчет и пройти защиту в соответствии требованиями работы.

*Планируемое время выполнения:*

Аудиторное время выполнения (под руководством преподавателя): 2 ч.

Время самостоятельной подготовки: 1 ч.

## Теоретическое введение

Одними только последовательными действиями в программировании не обойтись, поэтому при написании алгоритмов используется еще и ветвление.

Условная инструкция - основной инструмент выбора в Python определяет, какое действие следует выполнить в зависимости от значения переменных в момент проверки условия.

Операторы условий:

`==` (если равны оба операнда, то условие становится истинным),

`!=` (если не равны оба операнда, то условие становится истинным),

`>`, `<`, `>=`, `<=` (аналогично).

Для реализации выбора из нескольких альтернатив можно использовать конструкцию *if* – *elif* – *else*

Синтаксис инструкции *if*

- Обязательная часть *if* с условным выражением,
- далее могут следовать одна или более необязательных частей *elif*,
- и, наконец, необязательная часть *else*

*if* выражение\_1:

    инструкции\_(блок\_1)

*elif* выражение\_2:

    инструкции\_(блок\_2)

*elif* выражение\_3:

    инструкции\_(блок\_3)

*else*:

    инструкции\_(блок\_4)

**Важно:** Для выделения блоков кода используются исключительно отступ.

Логические строки с одинаковым размером отступа формируют блок, и заканчивается блок в том случае, когда появляется логическая строка с отступом меньшего размера

Пример:

```
a = int(input("Введите число:"))
```

```
if a < 0:
```

```
    print("Число отрицательное")
```

```
elif a == 0:
```

```
    print("Вы ввели ноль")
```

```
else:
```

```
    print("Ваше число положительное")
```

Логические операторы – союзы, которые позволяют объединять несколько условий.

**and** (логическое умножение) - возвращает True, если оба выражения равны True.

**or** (логическое сложение) - возвращает True, если хотя бы одно из выражений равно True.

**not** (логическое отрицание) - возвращает True, если выражение равно False.

Проверка истинности в Python:

Любое число, не равное 0, или непустой объект - истина.  
Числа, равные 0, пустые объекты и значение None – ложь.  
Операции сравнения и логические операторы возвращают True или False.

Литература:

[1] - глава 9. Стр. 57-65.

*Контрольные вопросы для самопроверки:*

1. В чем сущность "структуры выбора"?
2. Как организован выбор условий в программе?
3. Какие средства языка используются при описании структур выбора?

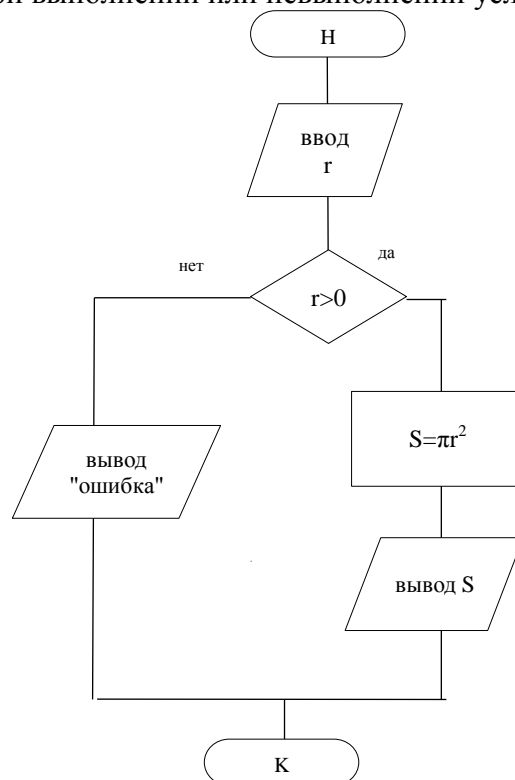
### Задание к лабораторной работе

Модифицировать схему алгоритма и программу предыдущей работы №1 в соответствии с заданием по своему варианту. Организовать разветвление, представить результаты или сообщение о невыполнении условия.

### Методические указания и порядок выполнения работы

Вычислить площадь круга по заданному радиусу с проверкой исходных данных: радиус должен быть положительным

Схема алгоритма добавлена блоком решения (проверкой аргумента на допустимость), с двумя выходами при выполнении или невыполнении условия:



Текст программы:

```
# Семенов А. 21-ИЭ-3
import math
r = float(input("Радиус: "))
if r > 0:
    p = math.pi * r**2
    print("Площадь круга: ", "%7.4f" % p)
else:
```

```
print (' Радиус должен быть >0')
```

Результаты выполнения:

```
Python 3.1.4 (default, Jun 12 2011, 15:05:44) [MSC v.1500 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> ===== RESTART =====
>>>
Радиус: 10
Площадь круга: 314.1593
>>> ===== RESTART =====
>>>
Радиус: -6
Радиус должен быть >0
```

### Индивидуальное задание

Номер варианта	Задача
0.	Определить в каком квадранте находится точка
1.	Определить является ли сумма чисел четной или нечетной
2.	Проверить исходные данные на допустимость
3.	Организовать проверку числа на количество цифр
4.	Организовать проверку числа на количество цифр
5.	Определить является ли треугольник равнобедренным
6.	Организовать проверку числа на количество цифр
7.	Проверить исходные данные на допустимость
8.	Проверить исходные данные на допустимость
9.	Определить наличие хотя бы одного положительного числа из трех
10.	Определить расположение точек: на оси или нет, все или одна
11.	Проверить исходные данные на допустимость
12.	Проверить исходные данные на допустимость
13.	Определить размещение центра окружности: на оси или нет
14.	Определить наличие цифры, введенной с клавиатуры, в числе

## 5. ЛАБОРАТОРНАЯ РАБОТА № 3. ИСПОЛЬЗОВАНИЯ СТРУКТУР ПОВТОРЕНИЯ. ЦИКЛЫ

### Общие сведения

#### Цель:

Получить навыки организации циклических вычислительных процессов, усвоить инструкции повторения Python

#### Материалы, оборудование, программное обеспечение:

Компьютерные классы с установленным высокоуровневым языком программирования Python.

#### Условия допуска к выполнению:

Изучить конспект по теоретической подготовке, теоретическое введение работы.

#### Критерии положительной оценки:

Для успешной сдачи лабораторной работы следует показать и объяснить программу, представить отчет в соответствии с требованиями и пройти защиту.

*Планируемое время выполнения:*

Аудиторное время выполнения (под руководством преподавателя): 2 ч.

Время самостоятельной подготовки: 1 ч.

## **Теоретическое введение**

Третьей необходимой алгоритмической конструкцией является цикл.

С помощью цикла можно описать повторяющиеся действия.

В Python имеются два вида циклов: цикл ПОКА (выполняется некоторое условие) и цикл ДЛЯ (всех значений последовательности).

Оператор цикла while выполняет указанный набор инструкций до тех пор, пока условие цикла истинно. Цикл while - цикл с предусловием.

Истинность условия и область выполняемых операторов определяется также как и в операторе if.

Синтаксис оператора while выглядит так.

while выражение:

инструкция\_1

инструкция\_2

...

инструкция\_n

Пример:

```
a = 0
```

```
while a < 7:
```

```
    print("A=", a)
```

```
    a += 1
```

Оператор цикла *for* выполняет указанный набор инструкций заданное количество раз, которое определяется количеством элементов в наборе. Параметры цикла должны быть целыми значениями.

Пример

```
for i in range(-5, 5, 1):
```

```
    print("Hello")
```

"Range" переводится как "диапазон", в котором может быть один, два или три аргумента.

Если задан только один, то генерируются числа от 0 до указанного числа, не включая его.

Если заданы два, то числа генерируются от первого до второго, не включая его.

Если заданы три, то третье число – это шаг.

*Литература:*

[1] - глава 9. Стр. 65-78.

*Контрольные вопросы для самопроверки:*

1. Что собой представляет алгоритмическая структура повторения? Какие основные операции она предусматривает?
2. Как в программе выделить область действия цикла?
3. Каким образом структура повторения "повторять пока" записывается в программах?
4. Что необходимо предусматривать для правильного исполнения структуры повторения?
5. В чем отличие операторов for и while?

### Задание к лабораторной работе

Организовать многократное (циклическое) выполнение предыдущей программы лабораторной работы № 2 при изменении параметра (или одного из исходных параметров) в заданном диапазоне. Представить схему алгоритма.

### Методические указания и порядок выполнения работы

В схеме предыдущей работы выделим участок (A), который должен повторяться при изменении радиуса.

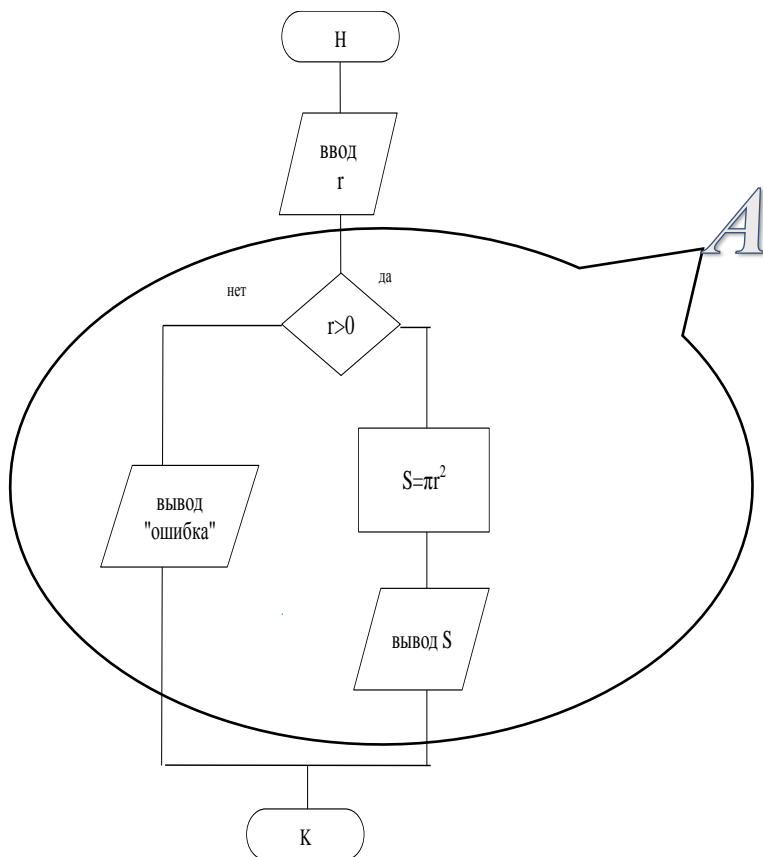
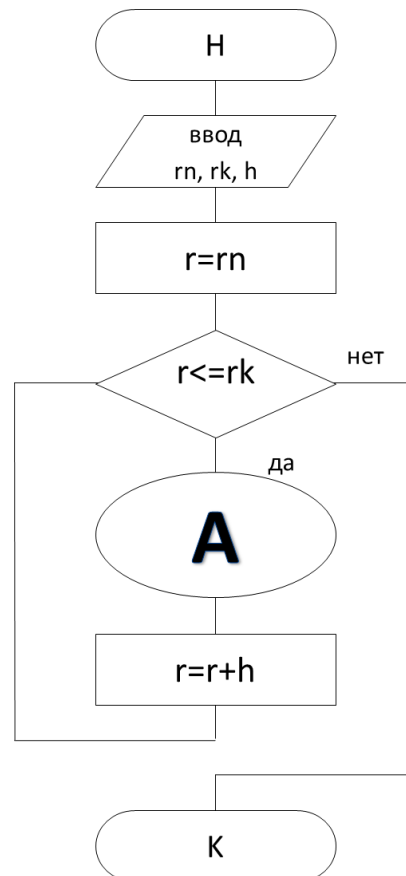




Схема цикла с предусловием содержит блоки, описанные в предыдущих работах:

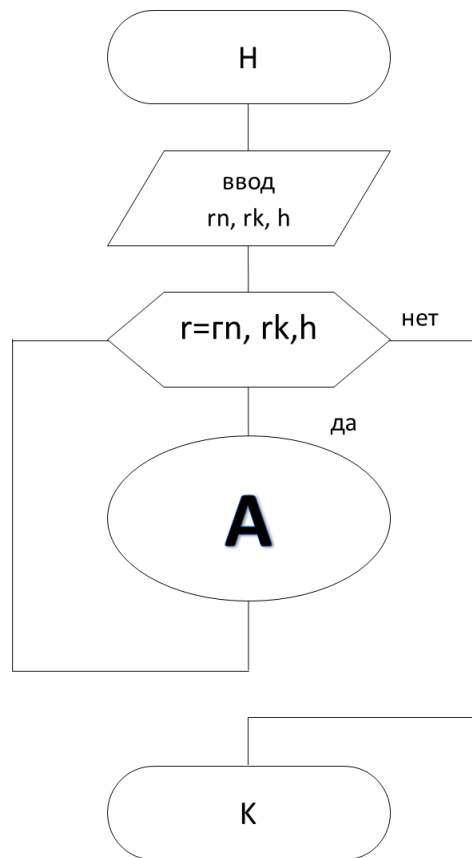


Программа и результат:

```
import math
print ("Введите диапазон и шаг изменения радиуса: ")
rn, rk, h = (map(float,input().split()))
r=rn
while r <= rk:
    print ("радиус= ", "%5.2f"%r, end="")
    if r>0:
        p= math.pi * r**2
        print (" Площадь круга:", "%7.2f"% p)
    else:
        print (" Радиус должен быть > 0 ")
    r+=h
```

```
Python 3.5.1 Shell
File Edit Shell Debug Options Window Help
Введите диапазон и шаг изменения радиуса:
-1 5 0.5
радиус= -1.00 Радиус должен быть > 0
радиус= -0.50 Радиус должен быть > 0
радиус= 0.00 Радиус должен быть > 0
радиус= 0.50 Площадь круга: 0.79
радиус= 1.00 Площадь круга: 3.14
радиус= 1.50 Площадь круга: 7.07
радиус= 2.00 Площадь круга: 12.57
радиус= 2.50 Площадь круга: 19.63
радиус= 3.00 Площадь круга: 28.27
радиус= 3.50 Площадь круга: 38.48
радиус= 4.00 Площадь круга: 50.27
радиус= 4.50 Площадь круга: 63.62
радиус= 5.00 Площадь круга: 78.54
>>>
```

Схема цикла for с параметром включает блок модификации, в котором указан диапазон (от rn до rk) и шаг повторения (h) переменной цикла (r):



Программа и результат:

```
import math
print ("Введите диапазон и шаг изменения радиуса: ")
rn, rk, h =map(int, input().split())
for r in range (rn,rk,h):
    print ("Радиус= ", "%7i"%r, " ", end="")
    if r>0:
        p= math.pi * r**2
        print (" Площадь круга: ", "%15.4f"%p)
    else:
        print (" Радиус должен быть положительным")
```

```
Python 3.5.1 Shell
File Edit Shell Debug Options Window Help
Введите диапазон изменения радиуса и шаг изменения:
-4 16 2
Радиус= -4 Радиус должен быть положительным
Радиус= -2 Радиус должен быть положительным
Радиус= 0 Радиус должен быть положительным
Радиус= 2 Площадь круга: 12.5664
Радиус= 4 Площадь круга: 50.2655
Радиус= 6 Площадь круга: 113.0973
Радиус= 8 Площадь круга: 201.0619
Радиус= 10 Площадь круга: 314.1593
Радиус= 12 Площадь круга: 452.3893
Радиус= 14 Площадь круга: 615.7522
>>> |
```

### Индивидуальное задание

В зависимости от данных варианта определить цикл (`while` – для любых данных, `for` – для данных целого типа). Организовать многократное (циклическое) выполнение предыдущей программы при изменении одного из исходных параметров в заданном диапазоне.

## 6. ЛАБОРАТОРНАЯ РАБОТА № 4. СПИСКИ. ОПЕРАЦИИ, МЕТОДЫ И ФУНКЦИИ СПИСКОВ

### Общие сведения

*Цель:*

Освоить понятия списки, операции и методы программирования при работе со списками. Научиться использовать функции и методы при программировании задач со списками.

*Материалы, оборудование, программное обеспечение:*

Компьютерные классы с установленным высокоуровневым языком программирования Python.

*Условия допуска к выполнению:*

Изучить конспект по теоретической подготовке, теоретическое введение работы.

*Критерии положительной оценки:*

Для успешной сдачи лабораторной работы следует показать и объяснить программу, представить отчет в соответствии с требованиями и пройти защиту.

*Планируемое время выполнения:*

Аудиторное время выполнения (под руководством преподавателя): 4 ч.

Время самостоятельной подготовки: 3 ч.

### Теоретическое введение

Для группировки множества элементов в Python используется список – `list`.

Список - индексированная последовательность значений, разделенных запятыми, заключенная в квадратные скобки.

```
s=[100, 200, 300, 400, 500, 45, -8, 402]
```

Для обращения к элементам списка используются индексы, которые представляют номер элемента в списке. Индексы начинаются с нуля. Для обращения к элементам с конца

можно использовать отрицательные индексы, начиная с -1. То есть у последнего элемента будет индекс -1, у предпоследнего -2 и так далее.

Для создания последовательного списка чисел можно воспользоваться функцией `range`, которая имеет три формы, описанные в предыдущей работе.

Например:

```
numbers = list(range(-10, 10, 2))
```

```
[-10, -8, -6, -4, -2, 0, 2, 4, 6, 8]
```

Или для генерации списков можно применять конструктор списков — цикл внутри квадратных скобок.

Пример:

```
s = [i*i for i in range(1,10)]
```

Конструктор может быть условным — список четных натуральных чисел:

```
a = [i for i in range(1,10) if i % 2 == 0]
```

```
[2, 4, 6, 8]
```

Элементы списка не обязательно должны быть одного типа:

```
lst = ['21-ВТ', 'КГТУ', 17, 2003]
```

Для списка можно получить срез:

```
lst[1:3] - ['КГТУ', 17]
```

Списки имеют произвольную вложенность, могут включать в себя любые вложенные списки:

```
student = [ ["Смирнов", 5], ["Федоров", 2], ["Алексеев", 4]]
```

Операторы

Создание копии списка

```
s2 = list(s1) или s3=s1[:]
```

Сложение и умножение списков

```
s4=s1+s3+s2
```

```
s5=s1*3
```

Оператор `in` используется для проверки наличия элемента в списке.

```
ball=[4, 5, 3, 2, 3, 4,...]
```

if 2 in ball:

```
    print ('Не все студенты сдали экзамен')
```

else:

```
    print ('Все студенты сдали экзамен')
```

Для работы с элементами списка используется итерация – организация цикла по элементам списка.

```
s = [100, 200, 300, 400, 500]
for k in range (len(s)):
    print ('Элемент: ', s[k], 'Индекс: ', k)
        Элемент: 100 Индекс: 0
        Элемент: 200 Индекс: 1
        Элемент: 300 Индекс: 2
        Элемент: 400 Индекс: 3
        Элемент: 500 Индекс: 4
```

или

```
for k in s:
    print ('Элемент: ', k)
        Элемент: 100
        Элемент: 200
        Элемент: 300
        Элемент: 400
        Элемент: 500
```

Методы и функции по работе со списками

Метод	Назначение
append (X)	Добавляет элемент X в конец списка
insert (индекс, X)	Добавляет элемент X в список по индексу
remove (X)	Удаляет элемент X. Удаляется только первое вхождение элемента. Если элемент не найден, генерирует исключение <code>valueerror</code>
clear ()	Удаление всех элементов из списка
index (X)	Возвращает индекс элемента X. Если элемент не найден, генерирует исключение <code>valueerror</code>
pop ([индекс])	Удаляет и возвращает элемент по индексу. Если индекс не передан, то просто удаляет последний элемент
count (X)	Возвращает количество вхождений элемента X в список
sort ()	Сортирует элементы. По умолчанию сортирует по возрастанию
reverse ()	Расставляет все элементы в списке в обратном порядке
Встроенные функции для работы со списками	
len (список)	Возвращает длину списка
sorted (список)	Возвращает отсортированный список
min (список)	Возвращает наименьший элемент списка
max (список)	Возвращает наибольший элемент списка

*Литература:*

[1] - глава 14, с. 123-128.

*Контрольные вопросы для самопроверки:*

8. Как можно определить список?
9. Какой метод определяет длину списка?
10. Элементами списка могут переменные разных типов?
11. Как определить наличие элементов в списке?
12. Какой метод предназначен для добавления элемента в список?

### **Задание к лабораторной работе**

Согласно своему варианту написать программу на языке Python.

Тестирование программы осуществить несколько раз, используя различные варианты исходных данных для получения всех предусмотренных в программе возможных вариантов результатов решения задачи.

Предусмотреть вывод сообщения при отрицательном результате решения.

Исходные списки следует генерировать (см. таблицу Модуль random).

### **Методические указания и порядок выполнения работы**

Пример.

Генерируется список 15 случайных целых чисел. Определить, сколько в нем положительных чисел, а сколько отрицательных.

Пояснение к задаче и алгоритм решения

1. Подключить модуль random. Создать пустой список, заполнить случайными числами в диапазоне от -10 до 10.
2. Ввести две переменные для подсчета положительных и отрицательных чисел. Присвоить им нули.
3. Перебираем элементы списка от первого до последнего.
4. Если очередное число - положительное, то увеличиваем на 1 переменную-счетчик положительных чисел, если отрицательное - увеличиваем на 1 переменную-счетчик отрицательных чисел.
5. При выводе результатов проверяем, изменились ли переменные, предназначенные для подсчета положительных и отрицательных чисел.

Программа:

```
# Соловьев П. 21-ИЭ-3
import random
a = []
for i in range(15):
    a.append (random.randint(0,10) )

print ('Исходный список: ', a)
positive = 0
negative = 0
for i in a:
    if i > 0:
        positive += 1
    if i < 0:
        negative += 1
if positive >0:
    print ("Положительных чисел: ", positive)
else:
    print ("Положительных чисел нет ")
```

```

if negative >0:
    print ("Отрицательных чисел: ", negative)
else:
    print ("Отрицательных чисел нет")

```

Возможные результаты:

```

Python Shell
File Edit Shell Debug Options Windows Help
>>>
Исходный список: [9, -3, 6, 2, -9, 1, -9, 4, 2, -9, -8, 1, 0, -4, 7]
Положительных чисел: 8
Отрицательных чисел: 6
>>> ===== RESTART =====
>>>
Исходный список: [9, 1, 2, 6, 10, 9, 3, 4, 2, 9, 4, 1, 0, 10, 7]
Положительных чисел: 14
Отрицательных чисел нет
>>> |
Ln: 13 Col: 4

```

### Методы модуля random

Метод	Значение	Пример	Результат
random.random()	случайное число от 0.0 до 1.0	random.random()	0.07500...
random.uniform (начало, конец)	случайное вещественное число в диапазоне от Начало до Конец	random.uniform(0, 20) random.uniform(0, 20)	15.3301818. 09232
random.randint (начало, конец)	случайное целое число в диапазоне от Начало до Конец	random.randint(1,27) random.randint(1,27)	9 22
random.choice (последовательность)	случайный элемент из строк, списков, кортежа	random.choice('Chewpa') random.choice([1,2,'a',8]) random.choice(['a','b'])	'h' 2 'a'
random.randrange (начало, конец, шаг)	Возвращает случайно выбранное число из последовательности	random.randrange(15,40,5)	25

### Индивидуальное задание

Номер варианта	Задача
0.	Организовать замену отрицательных чисел на минимальное значение списка, положительных элементов – на максимальное значение списка, нулевые значения заменить числом, введенным с клавиатуры
1.	Из исходного списка создать ещё два: положительные числа поместить в один список, а отрицательные - в другой
2.	Найти сумму элементов списка, кратных заданному числу, и произведение остальных
3.	Определить есть ли в исходном массиве два соседних положительных элемента. Найти первую и последнюю пару.
4.	Определить в списке наиболее встречаемое значение. Точно известно, что такой элемент один. Удалить из списка этот элемент
5.	Необходимо определить индексы элементов списка, значение которых не меньше заданного минимума и не больше заданного максимума

Номер варианта	Задача
6.	Нужно удалить из списка все элементы, которые больше $a$ и меньше $b$ , их значения вводятся с клавиатуры. При этом удаляемые числа нужно сохранить в другом списке
7.	Создать новый список из номеров элементов, равных заданному значению в исходном списке
8.	Сформировать новый список, состоящий номеров элементов, которые являются четными числами исходного списка
9.	Сформировать список, в котором элементы исходного списка, равные заданному значению, заменены на элементы с другим заданным значением
10.	Заполнить список с клавиатуры и вывести на экран в прямом и обратном порядке
11.	Определить индексы отрицательных элементов списка и их количество, в новом списке эти элементы заменить числом, равным количеству
12.	Создать список из элементов, значения которых превосходят по модулю заданное число, определить их количество
13.	Найти сумму четных элементов списка с четными индексами, создать список из этих элементов
14.	Заменить все четные элементы списка на их квадраты, нечетные удвоить, 0 заменить числом, введенным с клавиатуры

## 7. ЛАБОРАТОРНАЯ РАБОТА № 5. РАБОТА С МАТРИЦАМИ

### Общие сведения

#### *Цель:*

Освоить понятия матрицы, овладеть методами программирования при работе с элементами матриц.

#### *Материалы, оборудование, программное обеспечение:*

Компьютерные классы с установленным высокоуровневым языком программирования Python.

#### *Условия допуска к выполнению:*

Изучить конспект по теоретической подготовке, теоретическое введение работы.

#### *Критерии положительной оценки:*

Для успешной сдачи лабораторной работы следует показать и объяснить программу, представить отчет в соответствии с требованиями и пройти защиту.

#### *Планируемое время выполнения:*

Аудиторное время выполнения (под руководством преподавателя): 4 ч.

Время самостоятельной подготовки: 3 ч.

### Теоретическое введение

#### Матрицы

В языке программирования Python таблицу можно представить в виде списка строк, каждый элемент которого является в свою очередь списком, например, списком чисел.

8	14	10	45
17	22	6	77

Матрицы - двумерный массив или список строк, каждый элемент которого также список чисел, имеет два индекса  $i$  и  $j$ .



Например:

```
a = [ [8, 14, 10, 45], [17, 22, 6, 77] ]
```

Внешний список будет хранить строки матрицы, а внутренний – элементы этих строк.

a[0] – это [8, 14, 10, 45]

a[0][2] – это 10

Как правило, для работы с элементами матриц нужно использовать два цикла. Например, внешний цикл по номеру строки, внутренний цикл по элементам строки (или наоборот).

Например, вывести двумерный числовой список на экран построчно, можно так:

```
for i in range (len(a)):
    for j in range (len(a[i])):
        print ('%4d'% a[i][j], end=' ')
    print()
```

```
>>>
```

```
8 14 10 45
```

```
17 22 6 77
```

```
>>>
```

При создании вложенных списков рекомендуется использовать генератор, создав список из n элементов, каждый из которых будет списком, состоящих из m, например, нулей.

```
n=4
```

```
m=4
```

```
a = [[0] * m for i in range(n)]
```

Затем заполнить матрицу случайными числами:

```
import random
```

```
for i in range(n):
```

```
    for k in range(m):
```

```
        a[i][k]= (random.randint(-100,100))
```

Вывести результат:

```
for i in range (n):
```

```
    for j in range (m):
```

```
        print ('%6d'% a[i][j], end=' ')
```

```
    print ()
```

```
>>>
```

```
-74 -81 -24 -79
```

```
-49 90 54 -54
```

```
42 69 58 -95
```

```
-86 -59 53 85
```

```
>>>
```

*Литература:*

[1] - гл. 14, с. 123-128.

[3] - гл. 21, с. 49-60.

*Контрольные вопросы для самопроверки:*

1. Что такое матрица?
2. В чем сходство и отличие данных типа матрица и список?
3. Как организованы матрицы в Питоне?
4. Какие способы заполнения матрицы Вы знаете?
5. Какие типовые алгоритмы обработки матриц существуют?

## Задание к лабораторной работе

Согласно своему варианту написать программу на языке Python с выводом исходной матрицы.

Предусмотреть вывод сообщения при отрицательном результате решения.

Исходные матрицы генерировать, либо задавать в программе.

## Методические указания и порядок выполнения работы

Пример

Определить индексы минимального положительного элемента матрицы.

Пояснение к задаче и алгоритм решения

Подключить модуль random.

Ввести количество строк и столбцов матрицы (n, m).

Создать матрицу a, заполненную случайными числами в диапазоне от -100 до 100 и вывести на экран.

Определить элемент f=101 и его индексы (mi и mj) как искомые.

Организовать внешний цикл по номеру строки, внутренний цикл по элементам строки.

Если очередное число – положительное и меньше f, то запомнить число и индексы.

Вывести результат или сообщение об отсутствии положительных чисел в матрице.

Программа:

```
import random
```

```
n, m = map(int, input('Введите количество строк и столбцов матрицы: ').split())
```

```
#.....
```

```
# см. пример программы выше
```

```
f=101
```

```
mi=0
```

```
mj=0
```

```
for i in range (n):
```

```
    for j in range (m):
```

```
        if a[i][j]>0 and a[i][j]<f:
```

```
            f=a[i][j]
```

```
            mi=i
```

```
            mj=j
```

```
if f<100:
```

```
    print ("Минимальный положительный элемент матрицы: ", f, "[",mi,"][",mj,"]")
```

```
else:
```

```
    print ("В матрице нет положительных элементов")
```

Результат:

```
>>>
```

```
Введите количество строк и столбцов матрицы: 4 5
```

```
58  98  -95  18  -72
```

```
88  27  -83  13  89
```

```
-44 -46  99  -91  -72
```

```
-29  30  -26  49  -23
```

```
Минимальный положительный элемент матрицы: 13 [ 1 ][ 3 ]
```

или

```
Введите количество строк и столбцов матрицы: 4 5
```

```
-84 -64  -72  -89  -33
```

```
-17 -79  -92  -14  -92
```

-33 -49 -65 -92 -68  
 -35 -97 -15 -74 -44

В матрице нет положительных элементов

### Индивидуальное задание

Номер варианта	Задача
0.	Напишите программу определения номеров строк, все элементы которых ненулевые и вычислить сумму положительных элементов каждого столбца
1.	Напишите программу, которая вычисляет сумму отрицательных элементов в каждом столбце матрицы и количество отрицательных элементов в каждой строке матрицы
2.	Напишите программу, которая вычисляет суммы чётных элементов каждой строки матрицы и сумму нечетных каждого столбца
3.	Напишите программу, которая вычисляет среднее арифметическое значение положительных элементов массива и определяет индексы и количество нулевых элементов
4.	Напишите программу получения списка симметричных двухзначных чисел матрицы. Например, 17 и 71
5.	Напишите программу, которая выводит строку матрицы, сумма элементов которой наибольшая, и столбец с наибольшей суммой
6.	Напишите программу, которая определяет максимальный элемент каждого столбца матрицы и максимальный каждой строки
7.	Напишите программу, которая вычисляет среднее арифметическое значение отрицательных элементов и определяет индексы элементов, попадающих в интервал от 2 до 6
8.	Напишите программу, которая находит максимальный элемент матрицы, и удаляет из матрицы строку и столбец, на которых данный элемента расположен
9.	Напишите программу, которая выводит на экран строки матрицы, содержащие нули, и столбцы с нулевыми элементами
10.	Напишите программу получения списка двухзначных чисел матрицы, состоящих из одинаковых цифр. Например, 11 или 77
11.	Напишите программу, которая определяет минимальный элемент каждого столбца и каждой строки
12.	Напишите программу, которая вычисляет сумму положительных элементов каждого столбца и определяет количество строк, сумма элементов которых отрицательна
13.	Напишите программу получения из элементов матрицы трёх списков. Первый состоит из чисел, больших введенного значения, второй - из элементов, меньших этого числа, третий – из элементов, равных числу
14.	Напишите программу, которая определяет количество столбцов и строк матрицы, содержащих заданный элемент

## 8. ЛАБОРАТОРНАЯ РАБОТА № 6. СТРОКИ. ОПЕРАЦИИ, МЕТОДЫ И ФУНКЦИИ СТРОК

### Общие сведения

*Цель:*

Освоить методы программирования при работе с символьными данными в программе, изучить и использовать функции обработки текста.

*Материалы, оборудование, программное обеспечение:*

Компьютерные классы с установленным высокоуровневым языком программирования Python.

*Условия допуска к выполнению:*

Изучить конспект по теоретической подготовке.

*Критерии положительной оценки:*

Для успешной сдачи лабораторной работы следует показать и объяснить программу, представить отчет в соответствии требованиями работы и пройти защиту.

*Планируемое время выполнения:*

Аудиторное время выполнения (под руководством преподавателя): 4 ч.

Время самостоятельной подготовки: 3 ч.

### **Теоретическое введение**

Строка представляет последовательность символов, заключенных в кавычки. В Python используют как одинарные, так и двойные кавычки. Строки – это простой тип данных наряду с целыми и вещественными числами.

Строку рассматривают как упорядоченную последовательность элементов, т.е. к символам можно обращаться:

```
>>> name="Алексей"
```

```
>>> name[3]
```

```
'к'
```

```
>>>
```

Функция str() приводит числа к строке:

```
>>> str(4)
```

```
'4'
```

```
>>>
```

К базовым операциям относятся:

- конкатенация

```
>>> str(4)+ str(4)
```

```
'44'
```

```
>>>
```

- дублирование

```
>>> "МА"*2
```

```
'МАМА'
```

```
>>>
```

- сравнение

Строки (символы) могут участвовать в операциях сравнения. Тот факт, что буквы можно сравнивать, связан с тем, что в таблице кодов символов они идут друг за другом по порядку. Например, буква "в" имеет код на единицу меньше, чем "г", т.е. стоит перед "г".

Если первые символы одинаковые, в расчет берутся вторые символы при их наличии:

```
>>> "Петров" > "Петашевский"
```

```
True
```

```
>>>
```

При сравнении строк принимается во внимание символы и их регистр. Цифровой символ условно меньше, чем любой алфавитный символ. Алфавитный символ в верхнем регистре условно меньше, чем алфавитные символы в нижнем регистре. Зависимость от регистра не всегда желательна, в этом случае перед сравнением приводят обе строки к одному из регистров.

- Извлечение среза

Оператор извлечения среза: [X:Y]. X – начало среза, а Y – окончание. Символ с номером Y в срез не входит. По умолчанию первый индекс равен 0, а второй - длине строки.

```
>>> s = 'информация'
>>> s[2:7]
'форма'
>>> s = 'Мадам Заказ'
>>> s[::-1] # "Переворачивает текст"
'закаЗ мадаМ'
>>>
```

### Методы и функции по работе со строками

Функция или метод	Назначение
s[i]	Обращение по индексу
s[i:j:step]	Извлечение среза
len(s)	Длина строки
оператор членства in*	Возвращает истину, если элемент присутствует в строке, иначе возвращает ложь
оператор not in*	Возвращает истину, если элемента нет в строке
find(str[, start[, end]])	Поиск подстроки в строке. Возвращает номер первого вхождения (индекс подстроки в строке) или -1
rfind(str[, start[, end]])	Поиск подстроки в строке. Возвращает номер последнего вхождения или -1
index(str[, start[, end]])	Поиск подстроки в строке. Возвращает номер первого вхождения или вызывает ValueError
rindex(str[, start[, end]])	Поиск подстроки в строке. Возвращает номер последнего вхождения или вызывает ValueError
replace(old, new[, num])	Заменяет в строке S подстроку (old) на другую (new) num раз. Если num не указано, то заменить все вхождения. Если num = 1, то заменить первое вхождение. Если num = 3 - заменить первые 3 вхождения
split()	Разбивает строку на подстроки в зависимости от разделителя
join(список)	Объединяет строки в одну строку, вставляя между ними определенный разделитель
isdigit()	Возвращает True, если все символы строки - цифры
isnumeric()	Возвращает True, если строка представляет собой число
isalpha()	Возвращает True, если строка состоит только из алфавитных символов
isalnum()	Состоит ли строка из цифр или букв
islower() (isupper())	Возвращает True, если строка состоит только из символов в нижнем регистре. Знаки препинания и цифры дают True (в верхнем регистре)
istitle()	Возвращает True, если ВСЕ слова в строке начинаются с заглавной буквы
capitalize()	Переводит первый символ строки в верхний регистр, а все остальные в нижний
upper() (lower())	Переводит строку в верхний регистр (в нижний регистр)
title()	Начальные символы всех слов в строке переводятся в верхний регистр
count(str, [start],[end])	Возвращает количество вхождений подстроки в диапазоне (0 и

Функция или метод	Назначение
	длина строки по умолчанию)
lstrip([chars])	Удаляет начальные пробелы из строки
rstrip([chars])	Удаляет конечные пробелы из строки
swapcase()	Переводит символы нижнего регистра в верхний, а верхнего - в нижний
title()	Первую букву каждого слова переводит в верхний регистр, а все остальные в нижний

#### Литература:

1] - гл. 13, с. 110-120.

Контрольные вопросы для самопроверки:

1. Как можно определить строку?
2. Какой метод возвращает длину строки?
3. Как разбить строку по пробелам?
4. С помощью какого метода можно сформировать из списка строку?
5. Как организовать доступ к элементам строки?

#### Задание к лабораторной работе

Согласно своему варианту написать программу на языке Python.

Тестирование программы осуществить несколько раз, используя различные варианты исходных данных для получения всех предусмотренных в программе возможных вариантов результатов решения задачи (включая отрицательный).

В предложении слова разделять одним пробелом, знаки препинания писать слитно с предшествующим словом.

#### Методические указания и порядок выполнения работы

##### Пример

Посчитать количество русских строчных и прописных букв во введенной строке. Учитывать только английские буквы. При отсутствии букв в строке выдавать соответствующее сообщение.

Пояснение к задаче и алгоритм решения

- Ввести две переменные для подсчета строчных и прописных букв. Присвоить им нули.
- Перебрать символы строки от первого до последнего (номер последнего определяется длиной строки),
  - Если очередной символ - это буква, которая не меньше "а" и не больше "я", то увеличить на 1 переменную-счетчик строчных букв,
  - Иначе аналогично проверить принадлежность символа диапазону от "А" до "Я". Если символ ему принадлежит, то увеличить счетчик прописных букв.
  - Как видно из алгоритма, небуквенные символы вообще не учитываются.
  - Тот факт, что буквы можно сравнивать, связан с тем, что в таблице кодов символов они идут друг за другом по порядку. Например, буква "а" имеет код на единицу меньше, чем "б", т.е. стоит перед "б".
  - При выводе результатов проверить, изменились ли переменные, предназначенные для подсчета строчных и прописных букв

Программа:

```
s = input('Введите строку ')
let_s = 0
```

```

let_b = 0
for i in s:
    if 'a' <= i <= 'я':
        let_s += 1
    else:
        if 'A' <= i <= 'Я':
            let_b += 1
if let_s > 0:
    print ('кол-во строчных букв', let_s)
else:
    print ('нет строчных букв в предложении')
if let_b > 0:
    print ('кол-во прописных букв', let_b)
else:
    print ('нет прописных букв в предложении')

```

Результат:

### Индивидуальное задание

Номер варианта	Задание
0.	Посчитать количество гласных и согласных букв (строчных и прописных) во введенной строке
1.	Проверить в предложении соответствие числа открывающихся и закрывающихся круглых скобок
2.	Определить правильность регистра первой буквы слова после знаков препинания (.,!?). Внести исправления при неверном написании
3.	В символьной строке каждое слово перевернуть (например, "кот" на "ток")
4.	Определить, сколько букв введенного слова во введенной строке. Отметить вариант отсутствия и наличия всех букв
5.	Вывести арифметические знаки операций (+ - * / **), которых нет в строке. Учесть варианты отсутствия любого знака и наличия всех
6.	Определить, все ли гласные буквы встречаются в предложении без учета регистра и с учетом регистра
7.	Определить, есть ли в предложении слова, содержащие только цифры, и их количество
8.	Определить число слов в предложении, начинающихся согласными буквами, заменить все буквы этих слов на любой символ (не букву)
9.	Составить из букв введенного слова несколько любых их сочетаний (слов) любой длины. Каждую букву строки можно использовать неоднократно
10.	Определить какие логические операции содержит предложение. Предусмотреть варианты наличия и отсутствия всех операций
11.	В строке заменить пробелы звездочкой. Если встречается подряд несколько пробелов, то их следует заменить одним знаком "*", пробелы в начале и конце строки удалить. Вычислить общее количество пробелов и количество удаленных
12.	В исходной строке заменить буквы русского алфавита на латинские и наоборот, сохраняя регистр (использовать расположение букв на клавиатуре)

Номер варианта	Задание
13.	В исходной строке выделить пробелами заданную букву, подсчитать количество полученных слов
14.	Определить самое длинное слово в строке, вывести его номер (или номера)

## 9. ЛАБОРАТОРНАЯ РАБОТА № 7. ФУНКЦИИ. АРГУМЕНТЫ ФУНКЦИИ. ВЫЗОВ ФУНКЦИИ. ЛОКАЛЬНЫЕ И ГЛОБАЛЬНЫЕ ПЕРЕМЕННЫЕ

### Общие сведения

#### *Цель:*

Освоить назначение функции в программировании, научиться определять функции, назначение и описание параметров.

#### *Материалы, оборудование, программное обеспечение:*

Компьютерные классы с установленным высокоуровневым языком программирования Python.

#### *Условия допуска к выполнению:*

Изучить конспект по теоретической подготовке, теоретическое введение работы.

#### *Критерии положительной оценки:*

Для успешной сдачи лабораторной работы следует показать и объяснить программу, представить отчет в соответствии с требованиями и пройти защиту.

#### *Планируемое время выполнения:*

Аудиторное время выполнения (под руководством преподавателя): 4 ч.  
Время самостоятельной подготовки: 3 ч.

### Теоретическое введение

Существует множество встроенных в язык программирования функций.

Например, в Python: print(), input(), int(), float(), str()

Программист всегда может определять свои функции – пользовательские функции.

Функция в программировании представляет собой обособленный участок кода, который можно вызывать, обратившись к нему по имени, которым он был назван. Функции позволяют значительно сократить объем программного кода, сделать понятнее и эффективнее.

Преимущество функции:

- Функции придают программе структуру
- Польза функций в возможности многократного вызова одного и того же кода из разных мест программы
- Функции разделяют программу на обособленные части, каждая из которых выполняет свою конкретную задачу.

Определение функции

Функцию можно определить при помощи ключевого слова def, за которым должно следовать название функции и список её возможных формальных параметров в круглых скобках:

```
def fun(.....):
```

```
.....
```

Пример:

**#Описание**

```
def func():
```



```
print ("Вы используете функцию!")
#Вызов
func ()
Результат:
Вы используете функцию!!
```

#### Выполнение функции

Само определение функции не вызывает исполнения кода из тела функции. Код исполняется только при вызове функции.

Когда функция вызывается, поток выполнения программы переходит к ее определению и начинает исполнять ее тело. После того, как тело функции исполнено, поток выполнения возвращается в основной код в то место, где функция вызывалась. Далее исполняется следующее за вызовом выражение.

В языке Python определение функции должно предшествовать ее вызовам. Это связано с тем, что интерпретатор читает код строка за строкой и о том, что находится ниже, ему еще неизвестно. Поэтому если вызов функции предшествует ее определению, то возникает ошибка.

#### Передача аргументов функции

Конкретные значения, которые передаются в функцию при ее вызове, называются аргументами или параметрами. Пример: для функции `rectangle` аргументами являются числа, значения `k` и `l`.

```
def rectangle(a, b):
    s=a*b
rectangle(4,6)
rectangle(56,67)
rectangle(k,l)
```

Пример: при вызове функции `add` аргументы представлены различными вариантами:

```
def add(a, b):
    return a + b

print ( add(1, 2) )
print ( add(a = 2, b = 3) )
summa = add (b = 4, a = 5)
print (total)
```

В примере результатом функции является переменная `summa`. Это стандартный путь вызова функции в случае, если необходимо дальше использовать её результат.

#### Область видимости аргументов при использовании функций

Область видимости указывает, когда и где переменная может быть использована. При определении переменных внутри функции, эти переменные могут быть использованы только внутри этой функции. Когда функция заканчивается, значения переменных не передаются.

#### Локальные и глобальные переменные

Локальные переменные - переменные, объявленные в функции. Локальная область видимости значений переменных - отдельно взятая функция.

Переменные основной программы видны всей программе, известны и доступны всем функциям, они являются глобальными. К ним можно обратиться по имени и получить связанное с ними значение. При необходимости передачи значений переменных функции их следует объявить глобальными (`global`).

#### Возврат значений из функции. Оператор `return`

Функции могут передавать какие-либо данные из своих тел в основную ветку программы. Говорят, что функция возвращает значение. В Python выход из функции и передача данных в то место, откуда она была вызвана, выполняется оператором `return`. Допускается использование нескольких `return`, в том числе для раннего выхода из функции.

В таблице представлено взаимодействие программы и функции, параметров программы и функции, рассмотренные выше.

Действие	Основная программа	Функция	Пример
Определение	#Алексеев А. 21-ИЭ-1.	def fun(): .....	def add(a, b): s= a + b print (s)
Активация	<F5>	Вызов	fun ()
Область действия переменных	Глобальные переменные	Локальные переменные	
		Глобальные переменные	global t, y
Параметры	Фактические		nn(2004, 1, 'КГТУ')
		Формальные	def (god, kurs, univ)
Передача параметров из...	Имя функции (аргументы)		fan (a, s, d)
		Инструкция	return g, h

*Литература:*

[2]- гл. 2, с. 13-15.

*Контрольные вопросы для самопроверки:*

1. Как по тексту программы определить, какое значение возвращает функция?
2. Какие функции называются логическими? Зачем они нужны?
3. В чем отличие локальных параметров от глобальных?
4. Расположение функции в программе?

### **Задание к лабораторной работе**

Согласно своему варианту написать программу реализации функции на языке Python. Представьте результаты для диапазона (или списка) значений одного из параметров задания. Выводить исходные данные и результат.

Так, для функции определения максимального элемента списка, получить результаты для нескольких списков, например:

[ [1, 56, 7], [3, 8, 90, -13],[ ],...]

Результат:

В списке [1, 56, 7] максимальный элемент – 56

### **Методические указания и порядок выполнения работы**

Пример:

```
# Определить функцию вычисления площади прямоугольника,
# используя глобальные переменные
def rectangle():
    a = float(input("Ширина: "))
    b = float(input("Высота: "))
    global result
    result = a*b
rectangle()
print("Площадь: %.2f" % result)
```

Пример:

```
#Вычислить площади боковой поверхности цилиндра,
# используя оператор return
# Соловьев П. 21-ИЭ-3
def cylinder():
    r = float(input("Радиус? "))
    h = float(input("Высота? "))
    if r<=0 or h<=0:
        print ("Введены неверные значения")
        return
    side = 2 * 3.14 * r * h
    return side
```

```
Python Shell
File Edit Shell Debug Options Windows Help
Радиус? 4
Высота? 6
Площадь боковой поверхности цилиндра: 150.72
>>> ===== RESTART =====
>>>
Радиус? 4
Высота? -5
Введены неверные значения
Площадь боковой поверхности цилиндра: None
Ln: 13 Col: 43
```

Пример возврата нескольких значений:

```
# Вычисление площадей боковой поверхности и основания цилиндра
def cylinder():
    r = float(input("Радиус? "))
    h = float(input("Высота? "))
    side = 2 * 3.14 * r * h
    circle = 3.14 * r**2
    full = side + 2 * circle
    return side, full
sCyl, fCyl = cylinder ()
print ("Площадь боковой поверхности %.2f" % sCyl)
print ("Полная площадь %.2f" % fCyl)
```

```

Python Shell
File Edit Shell Debug Options Windows Help
>>>
Радиус? 5
Высота? 5
Площадь боковой поверхности 157.00
Полная площадь 314.00
>>>
Ln: 9 Col: 4

```

### Индивидуальное задание

Номер варианта	Задание
0.	Напишите функцию вычисления n-ой степени числа f ( $f^n$ ).
1.	Напишите функцию, которая вычисляет максимальное по модулю число из трех, не используя встроенную функцию max
2.	Напишите логическую функцию, которая проверяет корректность введенной даты (число и месяц)
3.	Напишите функцию, которая вычисляет минимальную нечетную цифру из любого числа
4.	Напишите функцию, которая “разворачивает” десятичную запись числа наоборот, например, из 123 получается 321, а из 3210 — 123
5.	Напишите логическую функцию, которая определяет, верно ли что введенное значение можно преобразовать в число (вещественное или целое)
6.	Напишите логическую функцию, которая определяет, верно ли что числа введены в порядке возрастания
7.	Напишите функцию, которая по заданной дате определяет время года
8.	Напишите функцию, которая определяет наибольший и наименьший элементы списка и их индексы
9.	Напишите логическую функцию, которая определяет, является ли число автоморфным, то есть квадрат этого числа заканчивается этим же числом, например, 6 (36) или 5 (25)
10.	Написать функцию вычисления факториала натурального числа
11.	Написать функцию определения количества положительных элементов списка до первого отрицательного
12.	Напишите функцию, которая определяет и выводит по числу название месяца или сообщение об ошибке
13.	Напишите логическую функцию, которая определяет для A, B и C возможность нахождения корней квадратного уравнения $Ax^2+Bx+C=0$
14.	Напишите функцию преобразования списка с сохранением положительных элементов, остальные элементы заменить нулями

## 10. ЛАБОРАТОРНАЯ РАБОТА № 8. СЛОВАРИ. ОБРАБОТКА ИСКЛЮЧЕНИЙ

### Общие сведения

#### Цель:

Получить навыки работы с данными Python, представленными в виде словаря, освоить и научиться использовать тип данных – исключения.

#### Материалы, оборудование, программное обеспечение:

Компьютерные классы с установленным высокоуровневым языком программирования Python.

*Условия допуска к выполнению:*

Изучить конспект по теоретической подготовке, теоретическое введение работы.

*Критерии положительной оценки:*

Для успешной сдачи лабораторной работы следует показать и объяснить алгоритм задачи, программу, представить отчет и пройти защиту в соответствии требованиями работы.

*Планируемое время выполнения:*

Аудиторное время выполнения (под руководством преподавателя): 4 ч.

Время самостоятельной подготовки: 3 ч.

## Теоретическое введение

Словарь

Словарь (dictionary) – неупорядоченный набор элементов, в котором доступ к элементу выполняется по ключу. Ключом является любой неизменяемый тип данных (число, строка). Ключ должен быть уникальным. Однако могут быть одинаковые значения у разных ключей.

Ключ и значение разделяются двоеточием. Пары друг от друга отделяются запятыми.

Элементы словаря заключаются в фигурные скобки:

```
dict = {ключ1: значение1, ключ2: значение2, ...}
```

Например:

```
d= {'1 пара': 'История', '2 пара': 'Математика', '3 пара': 'Программирование'}
```

При создании словарей можно использовать генераторы:

```
d = {a: a ** 3 for a in range (7)}
```

```
{0: 0, 1: 1, 2: 8, 3: 27, 4: 64, 5: 125, 6: 216}
```

или

```
s=['Первый', 'Второй', 'Третий', 'Четвертый']
```

```
d1= {h+1: s[h] for h in range (len(s))}
```

```
{1: 'Первый', 2: 'Второй', 3: 'Третий', 4: 'Четвертый'}
```

Для создания словаря из двух списков одинаковой длины (списка ключей и списка значений) используют функцию zip():

```
name=['21-ИЭ ', '21-ВТ ', '20-ИЭ ', '20-ВТ ']
```

```
number= [65, 58, 43, 38]
```

```
group=dict( zip(name,number))
```

```
{'21-ИЭ ': 65, '20-ВТ ': 38, '20-ИЭ ': 43, '21-ВТ ': 58}
```

К словарям можно применять операторы сравнения: == и !=.

Цикл for используется для прохода по ключам словаря:

```
for i in group:
```

```
    print ('В группе ', i, ' студентов: ', group [i])
```

```
В группе 21-ИЭ студентов: 65
```

```
В группе 20-ВТ студентов: 38
```

```
В группе 20-ИЭ студентов: 43
```

```
В группе 21-ВТ студентов: 58
```

Аналогично спискам можно получить значение по ключу: d[key].

Операции in и not in проверяют принадлежность ключа словарю.

Добавить элемент: d[key] = значение. Если указан ключ из словаря, элемент изменит значение.

Удалить элемент: del d[key].

Словарь – это изменяемый тип данных. Чтобы избежать нежелательного изменения основного словаря создают копию:

```
group1 = group или group2 = group.copy()
```

Метод `keys` возвращает ключи всех элементов. Метод `values` возвращает представление значений. Метод `items` возвращает представление всех пар из ключей и значений. Метод `clear` удаляет все элементы словаря. Метод `deercopy` создает полную копию словаря.

Можно извлекать ключи и значения:

```
for key, value in group.items():
    print (key, 'студентов – ', value)
```

Сортировать по ключам:

```
sorted( group)
```

`['20-ВТ ', '20-ИЭ ', '21-ВТ ', '21-ИЭ ']`

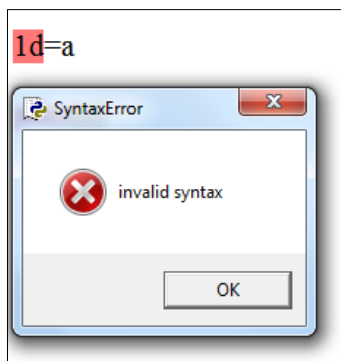
При сортировке по значениям требуется в качестве параметра указать значения словаря:

```
sorted (group.values())
```

`[38, 43, 58, 65]`

### Ошибки и исключения

Ошибки делятся на синтаксические и семантические. Синтаксические – погрешности в синтаксисе и пунктуации языка. Интерпретатор Python останавливает выполнение программы и выводит соответствующее сообщение, указав на место возникновения ошибки. Так действует обработчик ошибок.



Для языка Python здесь возникло исключение класса `SyntaxError`.

Нарушение семантики обычно означает, что при верно написанных выражениях с точки зрения синтаксиса языка, программа не работает так, как хотелось. В Python говорят об исключениях.

Исключения (exceptions) - ещё один тип данных в Python. Исключения необходимы для того, чтобы сообщать программисту об ошибках.

Например, вместо ввода числа вводится буква, или в ходе решения задачи возникает деление на ноль. В том и другом случае программа завершится с сообщением об ошибке.

Обрабатывать исключения можно при помощи оператора:

```
try..except (попытаться...исключение)
```

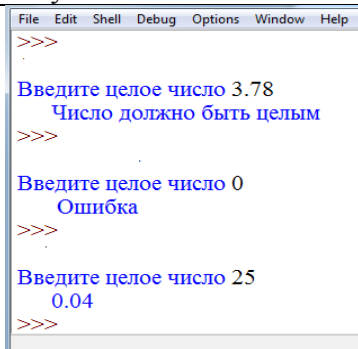
Оператор позволяет перехватывать возникающие исключения и обрабатывать их так, чтобы программа работала или корректно завершала свою работу. Конструкция похожа на условный оператор (`if...else`).

При этом все обычные команды помещаются внутрь `try`-блока, а все обработчики исключений – в `except`-блок.

Программа	Результат
<code>for s in range(-2,3):</code>	<code>-2 -0.5</code>
<code>try:</code>	<code>-1 -1.0</code>
<code>    k = 1/s</code>	<code>0 Ошибка</code>
<code>except ZeroDivisionError:</code>	<code>1 1.0</code>
<code>    k = " Ошибка"</code>	<code>2 0.5</code>
<code>print ("%4d"% s, " ", k)</code>	

Если в теле try исключения не возникает, то тело except не выполняется.

Но если в теле try возникнет еще какое-нибудь исключение, то него также следует указать отдельную ветку except.

Программа	Результат
<pre>try:     s=int (input("Введите целое число "))     k = 1/s except ZeroDivisionError:     k = " Ошибка" except ValueError:     k="Число должно быть целым" print ("    ", k)</pre>	

Несколько исключений можно сгруппировать:

```
except (ValueError, ZeroDivisionError):
```

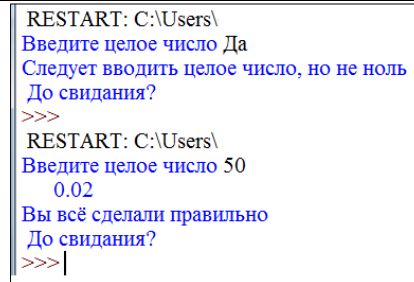
```
    k="Следует вводить целое число, но не ноль"
```

Ещё две инструкции, которые можно использовать при исключениях:

else и finally

Инструкция else выполняется в том случае, если исключения не было.

finally выполняет блок инструкций в любом случае, независимо от наличия или отсутствия исключений.

<pre>try:     s=int (input("Введите целое число "))     k = 1/s     print ("    ", k) except (ValueError, ZeroDivisionError):     print("Следует вводить целое число, но не ноль") else:     print ('Вы всё сделали правильно') finally:     print (' До свидания?')</pre>	
----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------

Полный список исключений представлен в ЭИОС

*Литература:*

[1] – гл. 15, с. 128-134.

*Контрольные вопросы для самопроверки:*

1. Что такое словарь?
2. В чем отличие словаря от списка?
3. Какие средства предназначены для создания словаря?
4. Какие элементы словаря должны быть уникальными?
5. Перечислите методы для работы со словарями.
6. Основное назначение использования обработчика исключений.

## Задание к лабораторной работе

Согласно своему варианту написать программу создания и обработки словаря на языке Python.

1. Создать два списка, каждый из 8-9 элементов. Первый список – ключи, уникальные элементы. Второй список – объекты. В качестве значений используйте элементы или списки.

2. Выполнить задания с выводом результат каждого пункта задачи:

- Преобразовать списки в словарь вида: {ключ1: объект1, ключ2: объект2,...}
- Получить значение элемента по ключу
- Добавить элемент в словарь
- Удалить элемент словаря
- Отсортировать словарь по ключам
- Создать копию словаря и отсортировать значения
- Создать список из ключей

3. В программе использовать хотя бы одну обработку исключений – при работе со словарем, или при вводе данных для запроса, или при отрицательном результате решения.

## Методические указания и порядок выполнения работы

Создать словарь, где ключ – фамилия студента, значение – оценки.

Определить сумму набранных баллов.

Сформировать новый словарь (ключ – фамилия, значение – сумма баллов).

Определить список отличников, удалить студентов, получивших все двойки.

В качестве объекта исключения используется проверка ключа.

```
fio=[ 'Алексеев','Логинов','Васильева','Голубев','Попова','Болтнев','Москалева']
ball= [[3,5,4],[2,2,2],[5,5,5],[4,4,4],[2,2,2],[5,5,4],[5,5,5]]
group=dict( zip(fio,ball))
print ('Результаты сессии')
for i in group:
    print ("{:11}\t{:7}".format( i, 'оценки: '), end="")
    print(group [i])
# Определяем списки для сумм баллов, отличников, двоечников
sball=[]
otl=[]
loser=[]
# Заполняем списки сумм баллов, отличников, двоечников
for i in group:
    s=sum(group [i])
    sball.append(s)
    if s==15:
        otl.append(i)
    if s==6:
        loser.append(i)

# Создаем словарь Фамилия: Сумма баллов
new_group=dict( zip(fio,sball))

# Проверяем наличие двоечников
if loser ==[]:
    print('Нет студентов, получивших все двойки')
else:
```



```

# Отчисляем студентов, получивших все двойки
for i in loser:
    print ('Отчислен: ', i)
    del new_group[i]
# Проверяем наличие отличников, выводим фамилии
if otl != []:
    print ('Отличники:', otl)
else:
    print ('В группе нет отличников')
# Выводим новый словарь и отсортированный список ключей (фамилий)
for i in new_group:
    print ("{:11}\t{:7}".format( i, 'сумма баллов: '), end="")
    print(new_group[i])
print ('Список группы: ',sorted( new_group))

# Исключение - проверка наличия Фамилии в списке ключей
# оператор raise передает имя ошибки/исключения
try:
    surname = input('Введите фамилию: ')
    if surname not in new_group:
        raise KeyError

except KeyError:
    print('Студент по ключу не найден')

else:
    print ('Есть такой студент')
Решение:
    Результаты сессии
    Голубев      оценки: [4, 4, 4]
    Алексеев    оценки: [3, 5, 4]
    Москалева   оценки: [5, 5, 5]
    Попова      оценки: [2, 2, 2]
    Болтнев     оценки: [5, 5, 4]
    Васильева   оценки: [5, 5, 5]
    Логинов     оценки: [2, 2, 2]
    Отчислен: Попова
    Отчислен: Логинов
    Отличники: ['Москалева', 'Васильева']
    Голубев     сумма баллов: 6
    Алексеев    сумма баллов: 12
    Москалева   сумма баллов: 6
    Болтнев     сумма баллов: 15
    Васильева   сумма баллов: 15
    Список группы: ['Алексеев', 'Болтнев', 'Васильева', 'Голубев', 'Москалева']
    Введите фамилию: Петров
    Студент по ключу не найден
    >>>
Другое решение:
    Результаты сессии
    Голубев     оценки: [4, 4, 4]
    Васильева   оценки: [5, 5, 5]

```

Москалева      оценки: [5, 5, 5]  
 Болтнев        оценки: [5, 5, 4]  
 Логинов        оценки: [2, 4, 2]  
 Попова         оценки: [2, 3, 32]  
 Алексеев      оценки: [3, 5, 4]  
 Нет студентов, получивших все двойки  
 Отличники: ['Васильева', 'Москалева']  
 Голубев        сумма баллов: 14  
 Васильева      сумма баллов: 15  
 Москалева      сумма баллов: 12  
 Болтнев        сумма баллов: 37  
 Логинов        сумма баллов: 15  
 Попова         сумма баллов: 8  
 Алексеев      сумма баллов: 12  
 Список группы: ['Алексеев', 'Болтнев', 'Васильева', 'Голубев', 'Логинов',  
 'Москалева', 'Попова']  
 Введите фамилию: Голубев  
 Есть такой студент  
 >>>

### Индивидуальное задание

Номер варианта	Вид словаря	Ключ	Значение (я)
0	Толковый словарь	Слово	Варианты объяснений
1	Компьютерный словарь	Термин	Варианты истолкований
2	Словарь синонимов	Слово	Варианты слов, близких по значению
3	Словарь антонимов	Слово	Варианты слов, различных по значению слова
4	Энциклопедический словарь (по любой теме)	Слово	Варианты истолкований
5	Переводной словарь (любой язык)	Слово	Перевод
6	Словарь стран	Страна	Сведения о стране
7	Словарь авторов	Автор	Произведения
8	Словарь месяцев	Месяц	Сведения о месяце
9	Словарь студентов	Фамилия	Сведения о студенте
10	Словарь городов	Город	Сведения о городе
11	Словарь имен	Имя	Варианты имен
12	Словарь типов данных в Python	Тип	Примеры данных
13	Словарь языков программирования	Язык	Примеры
14	Словарь направлений обучения	Направление	Сведения о направлении

## 11. ЛАБОРАТОРНАЯ РАБОТА № 9. ОСНОВНЫЕ МЕТОДЫ ДЛЯ РАБОТЫ С ФАЙЛАМИ

### Общие сведения

#### *Цель:*

Получить навыки работы с данными файла в Python, освоить чтение данных из файла и запись в файл.

#### *Материалы, оборудование, программное обеспечение:*

Компьютерные классы с установленным высокоуровневым языком программирования Python.

#### *Условия допуска к выполнению:*

Изучить конспект по теоретической подготовке, теоретическое введение работы.

#### *Критерии положительной оценки:*

Для успешной сдачи лабораторной работы следует показать и объяснить алгоритм задачи, программу, представить отчет и пройти защиту в соответствии требованиями работы.

#### *Планируемое время выполнения:*

Аудиторное время выполнения (под руководством преподавателя): 4 ч.

Время самостоятельной подготовки: 3 ч.

### Теоретическое введение

Файл - это набор данных внешней памяти с именем. Чтобы работать с сохранёнными данными, необходимы знания обработки файлов. Все операции с файлами состоят из трех частей: открытие файла, обработка данных и закрытие файла.

Python поддерживает множество различных типов файлов, но условно их можно разделить на два вида: текстовые и бинарные. Текстовый файл относится к файлам последовательного доступа.

Текстовые файлы – это, к примеру, файлы с расширением cvs, txt, html, т. е. любые файлы, которые сохраняют информацию в текстовом виде.

Бинарные файлы - это изображения, аудио и видеофайлы и т.д. В зависимости от типа файла работа с ним может немного отличаться.

Текстовые файлы содержат текст, разбитый на строки, символ перехода на новую строку (\n), символ возврата в начало строки (\r ).

Программы работают с файлом через файловую переменную. Общие методы для работы с файлами в Python:

- Открытие файла с помощью метода open()
- Закрытие файла методом close()
- Чтение файла с помощью метода read()
- Запись в файл посредством метода write()
- Перемещение указателя по файлу методом seek()

#### Атрибуты файлового объекта

Как только файл был открыт и появился файловый объект, можно получить следующую информацию о нем:

Метод	Назначение	Программа	Результат
file.name	Возвращает имя файла	myf = open ('primer.txt', 'w')	
file.closed	Возвращает		

Метод	Назначение	Программа	Результат
	True, если файл закрыт	print ("Имя файла: ", myf.name)	Имя файла: primer.txt
file.mode	Возвращает режим доступа, с которым был открыт файл	print ("Файл закрыт: ", myf.closed)  print ("Режим чтения или записи: ", myf.mode)	Файл закрыт: False Режим чтения или записи: w

Метод open()

Метод open() возвращает ссылку на файл, представляющий файловый объект, чтобы получить доступ к чтению, записи или добавлению.

Например:

```
f1 = open('work.txt', 'w')
```

```
f = open('D:/test.txt', 'r')
```

Синтаксис метода open()

```
my_file = open(имя_файла [, режим_доступа])
```

имя\_файла: строка, содержащая имя файла с расширением

режим\_доступа: строка, указывает, для чего открывается файл: для чтения, записи, добавления информации, и т.д.

Обычно используются режимы чтения ('r') и записи ('w').

Если файл открыт в режиме чтения, то запись в него невозможна. Можно только считывать данные из него.

Если файл открыт в режиме записи, то в него можно только записывать данные, считывать нельзя.

Если файл открывается в режиме 'w', то все данные, которые в нем были до этого, стираются. Файл становится пустым.

Если не надо удалять существующие в файле данные, тогда следует использовать вместо режима записи, режим дозаписи 'a'.

Если файл отсутствует, то открытие его в режиме 'w' создаст новый файл.

Бывают ситуации, когда надо гарантировано создать новый файл, избежав случайной перезаписи данных существующего.

В этом случае вместо режима 'w' используется режим 'x'. В нем всегда создается новый файл для записи.

Если указано имя существующего файла, то будет выброшено исключение. Потери данных в уже имеющемся файле не произойдет.

Метод close()

После завершения работы с файлом его обязательно нужно закрыть методом close().

Данный метод освободит все связанные с файлом используемые ресурсы.

Один и тот же файл можно открывать и закрывать в программе несколько раз с разными целями.

Метод read()

Синтаксис метода read()

```
my_file.read([count])
```

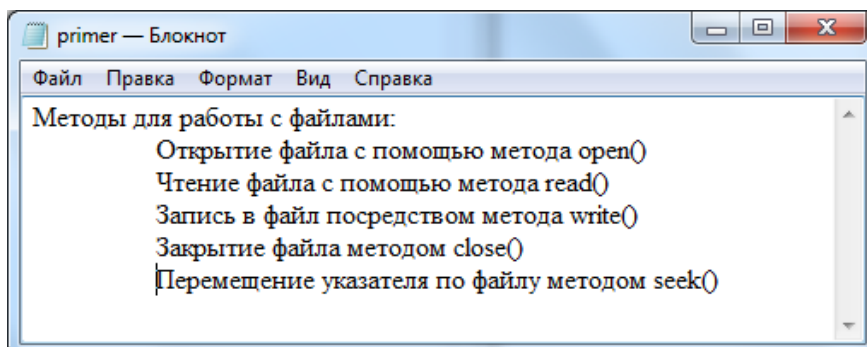
Метод читает информацию с начала файла и, если параметр count не указан, до конца файла.

Необязательный параметр count — это количество байт, которые следует прочитать из открытого файла.

Метод read() возвращает строку, конец строки считывается как '\n'.

Для того чтобы избежать проблем с путями до файлов в Windows используйте в них слэш '/', а также перед открывающей кавычкой пути файла ставьте букву u, указывающую на то, что строка в кодировке Unicode.

Пример 1: Чтение записей файла



<pre>filename = open ('primer.txt' , 'r') a=True print ('Тестовый файл ') while a:     line = filename.readline()     # читаем 1 строку     print (line)     if not line:         a=False filename.close()</pre>	<p>Тестовый файл Методы для работы с файлами:</p> <ul style="list-style-type: none"> <li>Открытие файла с помощью метода open()</li> <li>Чтение файла с помощью метода read()</li> <li>Запись в файл посредством метода write()</li> <li>Закрытие файла методом close()</li> <li>Перемещение указателя по файлу методом seek()</li> </ul>
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Пример 2: Чтение записей файла до определенной позиции

<pre>fd = filename.read(8) print (fd)</pre>	<p>Методы д</p>
---------------------------------------------	-----------------

Пример 3: Чтение записей файла может быть организовано следующим образом:

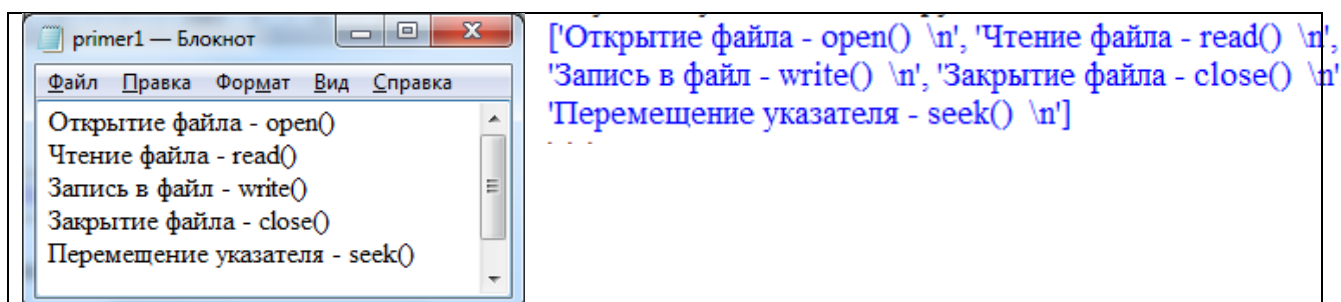
```
for line in open ("primer.txt", 'r'):
    print (line)
```

или:

```
for i in open("primer.txt"):
    print (i)
```

Пример 4: Метод readlines() читает строки файла и помещает их в список.

```
line = filename.readlines()
print (line)
```



При работе с файлами применяют оператор with. При его использовании нет необходимости закрывать файл, при завершении работы с ним эта операция будет выполнена автоматически.

Пример 5:

<pre>File Edit Format Run Options Window Help  with open ("my_file.txt", "r") as my_f:     for line in my_f:         print (line)  print ("Файл закрыт: ", my_f.closed) print ("В каком режиме файл открыт: ", my_f.mode)</pre>	<p>Открытие файла - open()</p> <p>Чтение файла - read()</p> <p>Запись в файл - write()</p> <p>Файл закрыт: True В каком режиме файл открыт: r &gt;&gt;&gt;  </p>
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Пример 6:

<pre>File Edit Format Run Options Window Help  # Пример представления результата вычислений x=10 y=5 f = open ("my_file.txt","w" )  f.write ("Вычисляем: \n")  f.write ( "{:d} + {:d} = {:d}\n".format(x, y, x+y) ) f.write ( "{:d} - {:d} = {:d}\n".format(x, y, x-y) ) f.write ( "{:d} * {:d} = {:d}\n".format(x, y, x*y) ) f.write ( "{:d} ** {:d} = {:d}\n".format(x, y, x**y) ) f.close () f=open ("my_file.txt", "r") for line in f:     print (line)</pre>	<p>Вычисляем:</p> <p>10 + 5 = 15</p> <p>10 - 5 = 5</p> <p>10 * 5 = 50</p> <p>10 ** 5 = 100000</p> <p>&gt;&gt;&gt;  </p>
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	-------------------------------------------------------------------------------------------------------------------------

Записать данные в файл можно с помощью метода write(), который возвращает число записанных символов.

В Python запись в файл построчно осуществляется с помощью записи нужной строки с последующей записью символа перевода строки '\n'.

Например:

```
grf = open ('list_gr.txt', 'w')
grf.write('18-ЗИЭ - 1 \n18-ЗИЭ - 2\n')
```

Каждый элемент будет записан в новой строке.

Пример записи построчно списка:

```
gr19 = ['19-ЗИЭ - 1', '19-ЗИЭ-2','19-ЗИЭ-3']
for g in gr19:
    grf.write(g+'\n' )
```

В цикле проходим по всем элементам списка и с помощью команды write записываем. Чтобы каждая запись была с новой строки, добавляем символ перевода строки.

Так же можно воспользоваться функцией writelines. Если передать в качестве ее параметра список, то она запишет элементы друг за другом в одну строку. Но можно поступить следующим образом: в качестве параметра передать генератор следующим образом:

```
gr20 = ['20-ЗИЭ - 1', '20-ЗИЭ - 2']
with open("list_gr.txt", "a") as file:
    file.writelines("%s\n" % line for line in gr20)
```

Пример 7: Заполнение файла данными

<pre>File Edit Format Run Options Window Help # Открываем файл для записи, заносим данные grf = open('list_gr.txt', 'w') grf.write('18-ЗИЭ - 1 \n18-ЗИЭ - 2\n')  # Включаем в файл список gr19 = ['19-ЗИЭ - 1', '19-ЗИЭ - 2', '19-ЗИЭ - 3'] for g in gr19:     grf.write(g+'\n') grf.close() print ("Список групп \n2019/20 года\n") grf = open ('list_gr.txt', 'r') print (grf.read())  # Добавляем в существующий файл новые данные gr20 = ['20-ЗИЭ - 1', '20-ЗИЭ - 2'] with open("list_gr.txt", "a") as file:     file.writelines("%s\n" % line for line in gr20) grf.close() print ("Список групп \n2020/21 года\n") grf = open ('list_gr.txt', 'r') print (grf.read())  Ln: 22 Col: 0</pre>	<p>Список групп 2019/20 года</p> <p>18-ЗИЭ - 1 18-ЗИЭ - 2 19-ЗИЭ - 1 19-ЗИЭ - 2 19-ЗИЭ - 3</p> <p>Список групп 2020/21 года</p> <p>18-ЗИЭ - 1 18-ЗИЭ - 2 19-ЗИЭ - 1 19-ЗИЭ - 2 19-ЗИЭ - 3 20-ЗИЭ - 1 20-ЗИЭ - 2</p>
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Помните, что создать текстовый файл с исходными данными можно в обычном блокноте.

Метод seek() - это встроенный метод в Python, он используется для установки текущей позиции в файле (или указателя файла).

К дополнительным методам для работы с файлами можно отнести метод tell(), который возвращает текущую позицию "условного курсора" в файле.

Пример 8:

```
File Edit Format Run Options Window Help
# Открываем файл для чтения, записываем текст,
# определяем количество символов, закрываем файл
myfile = open("primer.txt", "w")
res = myfile.write ("Изучаем Python. Скоро сдавать экзамен!!")
print (res, " - байт записано в файл")
myfile.close()

# Читаем и выводим текст файла
myfile = open ("primer.txt", "r")
print ("Текущая позиция при открытии файла - ",myfile.tell())
print ("Содержимое файла...")
print (myfile.read())
print ("Текущая позиция после прочтения файла - ",myfile.tell())
print ()

# Устанавливаем указатели на различные позиции в файле и выводим
print ("    Содержимое файла с 8-й позиции")
myfile.seek(8)
print ("    Текущая позиция при установки указателя - ",myfile.tell())
print (myfile.read())
print ("    Содержимое файла с 0-й позиции")
myfile.seek(0)
print (myfile.read())
print ("    Содержимое файла с 16-й позиции")
myfile.seek(16)
print (myfile.read())

Ln: 11 Col: 0
```

*Литература:*

[2] - гл. 4, с. 32-34.

*Контрольные вопросы для самопроверки:*

1. Что такое файл? Какие типы файлов используют в программировании?
2. Допустимы ли различные типы данных для элементов одной записи?
3. Какие основные действия с файлами? Для каких целей можно открыть файл?
4. Указать операторы: запись данных в файл и чтение данных из файла?
5. Что происходит с файлами, когда программа завершает свою работу?



## Задание к лабораторной работе

1. Создать блокнот из 5 – 7 записей с данными по варианту.
2. Написать программу, которая:
  - выводит на экран содержимое файла
  - выполняет вычисления в соответствии с заданием по варианту
  - создает новый файл в соответствии с заданием по варианту (предусмотреть вариант отсутствия данных по заданию)
  - выводит на экран содержимое нового файла
3. В программе использовать хотя бы одну обработку исключений при работе с файлом.

## Методические указания и порядок выполнения работы

Задание:

В текстовый файл построчно записаны фамилии студентов, наименование группы, процент правильных ответов при тестировании.

Вывести на экран и в новый файл список студентов с результатом тестирования менее 50% и посчитать средний результат тестирования.

Программа (в качестве исключения – проверка наличия файла):

```
file='result.txt'
try:
    f = open(file, 'r')
except FileNotFoundError:
    print("Файл не найден.")
    exit(0)

# Открываем файл для записи
ff = open ('total.txt', 'w')
# stud - для списка студентов, не прошедших тестирование
# grade_sum и count - для вычисления среднего значения
stud= []
grade_sum = 0
count = 0
print ('Исходный файл')

for line in f.readlines():
    surname, group, grade = line.split() # построчное чтение файла с обработкой данных
    print (surname, group, grade) # фамилия, группа, процент

    grade = int(grade[0:2]) # выделение числа
    grade_sum += grade
    count += 1
    if grade < 50:
        stud.append(surname) # получение списка
f.close()
```

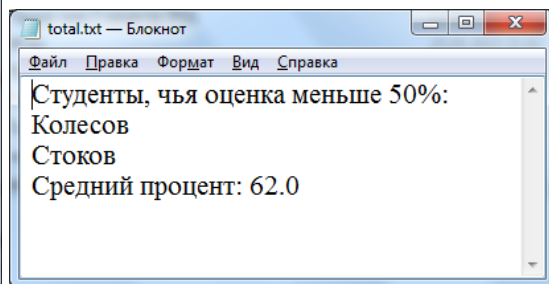
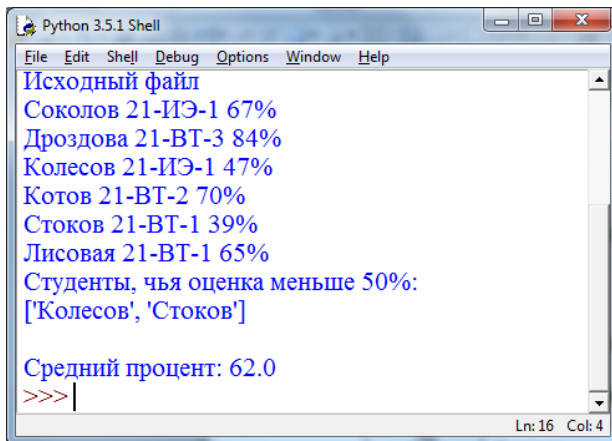
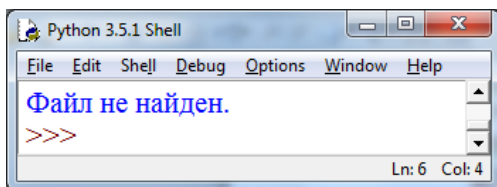
```

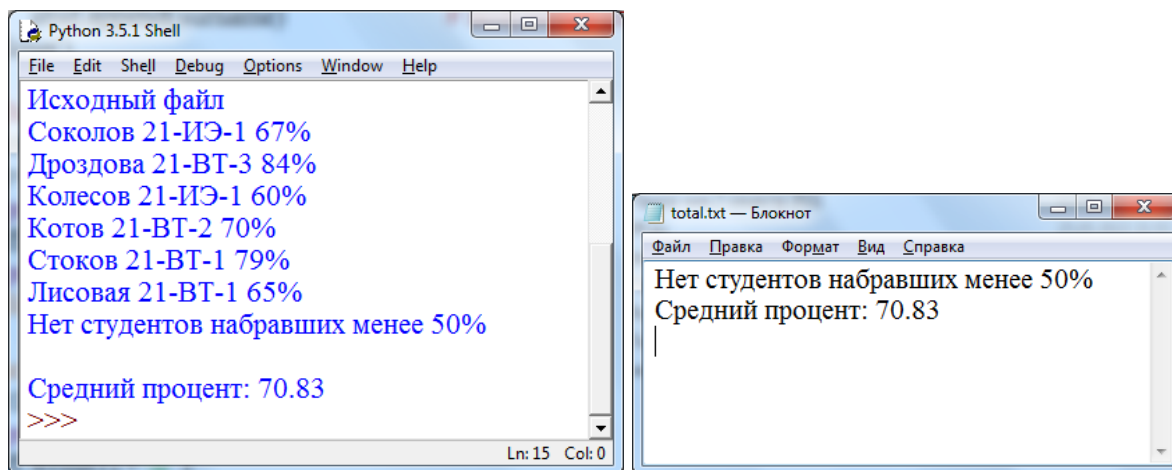
# Вывод результатов и заполнение нового файла
if stud:
    t="Студенты, чья оценка меньше 50%:"
    print (t)
    ff.write(t+'\n' )
    print(stud, sep='\n')
    for g in stud:
        ff.write(g+'\n' )
else:
    t1="Нет студентов, набравших менее 50%"
    print(t1)
    ff.write(t1+'\n' )

print()
t2="Средний процент: "+str(round( grade_sum / count,2))print (t2)
ff.write(t2+'\n' )
ff.close()

```

Возможные результаты:





### Индивидуальное задание

Номер варианта	Запись	Определить	Данные нового файла
0.	Группа Количество студентов Количество неудовлетворительных оценок за экзамен	Группу с самой плохой успеваемостью	Список групп, студенты которых сдали все экзамена
1.	Фамилия спортсмена Вид спорта Место в соревнованиях	Спортсмена, занявшего последнее место	Список спортсменов, завоевавших первые места
2.	Предложение со знаками препинания	Количество символов в тексте, которые не являются буквами	Список, состоящий только из слов в предложениях
3.	Название дисциплины Семестр обучения Вид отчетности (экзамен или зачет)	Количество экзаменов и количество зачетов	Список дисциплин в указанном семестре
4.	Наименование риэлтерской компании Количество проданных квартир Вырученная сумма	Самую успешную компанию	Список компаний, упорядоченный по выручке
5.	Фамилия студента Группа Экзаменационная оценка	Средний балл за экзамен	Список сдавших экзамен студентов
6.	Страна Стоимость тура Месяц	Страну с максимальной стоимостью тура	Список стран с турами в указанный месяц
7.	Предложение, содержащее числа	Сумму чисел	Предложения без чисел
8.	Фамилия сотрудника Должность Размер заработной платы	Самую высокооплачиваемую должность	Список сотрудников, чей оклад меньше указанной суммы
9.	Наименование страхового полиса Год окончания действия Страховая сумма	Наименование полиса с самой большой страховой суммой	Список полисов, срок действия которых заканчивается в указанном году

Номер варианта	Запись	Определить	Данные нового файла
10.	Фамилия студента Год поступления Сумма баллов ЕГЭ	Максимальный балл ЕГЭ	Список фамилий студентов, поступивших в указанном году
11.	Наименование направления Стоимость обучения Количество студентов	Наименование направления с самой большой стоимостью обучения	Список направлений, упорядоченный по стоимости обучения
12.	Наименование дисциплины Количество часов Семестр обучения	Самую трудоемкую дисциплину	Список дисциплин в указанный семестр
13.	Наименование вуза Год выпуска Количество выпускников	Количество выпущенных специалистов	Список наименований вузов и количество выпускников, упорядоченный по годам выпуска
14.	Фамилия клиента Количество заказов Стоимость заказов	Фамилию клиента с самым большим количеством заказов	Списки клиентов, отсортированные по стоимости заказов

## 12. ЛАБОРАТОРНАЯ РАБОТА № 10. ГРАФИКА В PYTHON

### Общие сведения

#### *Цель:*

Освоить методы программирования Python для формирования изображения с использованием графических примитивов.

#### *Материалы, оборудование, программное обеспечение:*

Компьютерные классы с установленным высокоуровневым языком программирования Python.

#### *Условия допуска к выполнению:*

Изучить конспект по теоретической подготовке и теоретическое введение.

#### *Критерии положительной оценки:*

Для успешной сдачи лабораторной работы следует показать и объяснить программу, представить отчет в соответствии требованиями работы и пройти защиту.

#### *Планируемое время выполнения:*

Аудиторное время выполнения (под руководством преподавателя): 2 ч.

Время самостоятельной подготовки: 3 ч.

### Теоретическое введение

Графический примитив — это простой или низкоуровневый объект, а также элементарная операция, с помощью которой можно выстроить более сложные объекты и реализовывать операции более высокого уровня.

Примитивы в графическом редакторе — это базовые элементы, такие как линии, кривые и полигоны, которые могут быть объединены для создания более сложных графических изображений. В программировании это основные операции, поддерживаемые

языком программирования. Для создания любого чертежа в компьютере такие графические примитивы - это то, что образует часть программного обеспечения.

## Модуль Graph

Графические интерфейсы можно создать с помощью библиотеки Tkinter.

Установочный файл интерпретатора Питона обычно включает пакет tkinter в составе стандартной библиотеки.

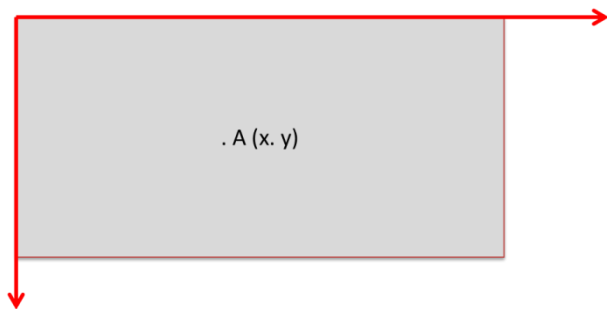
Модуль graph – это набор функций, который представляет собой «обёртку» для создания учебных графических программ на языке Python на основе виджета Canvas библиотеки Tkinter. (скачать модуль graph и файлы для его установки можно в ЭОИС в разделе данной практической работы или в Интернете на официальном сайте языка Python).

Обязательные операторы программы:

`from graph import *` (импортирует все функции модуля graph).

`run()` (функция запускает режим рисования и открывает графическое окно. Эта команда всегда должна завершать программу с использованием graph)

Поле для рисования называется холстом, его размеры совпадают с графическим окном. Холст – это растровый рисунок, каждая точка имеет координаты x и y.



В таблицах описаны функции организации окон, команды для рисования геометрических фигур, пояснения к которым представлены в примерах.

Действие	Функция	Пример
Объект главного окна	<code>mainWindow ()</code>	
Ширина и высота рабочей области окна	<code>windowSize (width, height)</code>	<code>windowSize (800, 600)</code>
Объект области рисования (холст)	<code>canvas ()</code>	
Начало координат (x, y) области рисования	<code>canvasPos (x, y)</code>	
Ширина и высота области рисования	<code>canvasSize (width, height)</code>	<code>canvasSize (700, 500)</code>
Диапазоны математической декартовой системы координат	<code>viewCoords (x1, x2, y1, y2)</code>	
Запускает основной цикл обработки сообщений; последняя строчка	<code>run ()</code>	
Закрывает графическое окно	<code>close ()</code>	
True, если точка с координатами (x, y) находится в пределах области рисования	<code>pointInView (x, y)</code>	<code>print (pointInView (980, 80))</code>

Действие	Функция	Пример
True, если окружность с центром в точке (x, y) радиуса r находится в пределах области рисования	circleInView (x, y, r)	

Команды рисования геометрических фигур		
Действие	Функция	Пример
Толщина пера	penSize (width)	penSize (5)
Цвет пера (см. Таблицу HTML цветов)	penColor (color) penColor (r, g, b)	penColor ("blue") penColor (178, 34, 34)
Цвет заливки (см. Таблицу HTML цветов)	brushColor (color) brushColor (r, g, b)	
Точка цвета с координатами (x, y)	point (x, y) point (x, y, color)	point (100, 100, 'red')
Перейти в точку, заданную координатами (x, y) или кортежем pos= (x, y) из этих координат	moveTo (pos) moveTo (x, y)	pos=(120,45) moveTo (pos)
Линия из текущего положения в точку, с координатами (x, y) или кортежем pos= (x, y) из этих координат	lineTo (pos) lineTo (x, y)	
Линия между точками (x1, y1) и (x2, y2)	line (x1, y1, x2, y2)	
Ломаная линия, составленная из точек, представленных списком кортежей координат	polyline (p)	d=[ (80,30), (130,30), (130,160), (80,80)] polyline (d)
Многоугольник (замкнутая ломаная линия)	polygon (points)	polygon(d)
Прямоугольник с координатами противоположащих углов (x1, y1) и (x2, y2)	rectangle (x1, y1, x2, y2)	rectangle (10, 20, 60, 50)
Окружность с центром в точке (x, y) радиусом r	circle (x, y, r)	circle (200, 100, 50)
Овал (вписанный в прямоугольнике с точками (x1, y1) и (x2, y2))	oval (x1, y1, x2, y2)	oval (100, 50, 400, 200)
Текст с точки (x, y)	label ("Текст", x, y)	label ("Учимся рисовать", 50, 50, font='20')
Рисунок из файла (загружаются рисунки формата GIF)	image (x, y, fileName)	image (50, 50, 'c:/LP.gif')

Таблица HTML цветов		
Формат Color	HEX – шестнадцатеричный формат	RGB – формат Красный, Зеленый, Голубой
<b>Black</b>	<b>#000000</b>	<b>0, 0, 0</b>
Gray	#808080	128, 128, 128
Silver	#C0C0C0	192, 192, 192
<b>White</b>	<b>#FFFFFF</b>	<b>255, 255, 255</b>
<b>Fuchsia</b>	<b>#FF00FF</b>	<b>255, 0, 255</b>
<b>Purple</b>	<b>#800080</b>	<b>128, 0, 128</b>
<b>Red</b>	<b>#FF0000</b>	<b>255, 0, 0</b>
<b>Maroon</b>	<b>#800000</b>	<b>128, 0, 0</b>
<b>Yellow</b>	<b>#FFFF00</b>	<b>255, 255, 0</b>
Olive	#808000	128, 128, 0
<b>Lime</b>	<b>#00FF00</b>	<b>0, 255, 0</b>
<b>Green</b>	<b>#008000</b>	<b>0, 128, 0</b>
<b>Aqua</b>	<b>#00FFFF</b>	<b>0, 255, 255</b>
Teal	#008080	0, 128, 128
<b>Blue</b>	<b>#0000FF</b>	<b>0, 0, 255</b>
<b>Navy</b>	<b>#000080</b>	<b>0, 0, 128</b>

*Литература:*

[2] - гл. 12, с. 73-104.

*Контрольные вопросы для самопроверки:*

1. Какие средства языка предназначены для изображения простейших фигур?
2. Как задать цвет и толщину линий?
3. Для каких фигур можно выполнить заливку и как задать цвет заливки?
4. Как представлены координаты точек на холсте?

### **Задание к лабораторной работе**

Согласно своему варианту написать программу на языке Python, формирующую указанное изображение с использованием графических примитивов. Выбор программного обеспечения предоставляется студенту (модуль Graph или компонента Canvas). В рисунке использовать минимум шесть различных простейших фигур. Цветовую гамму и размер выбрать самостоятельно. **Допускается собственный рисунок.**

Написать программу для построения графика функции

## Методические указания и порядок выполнения работы

Пример 1.

Нарисовать флаг и шарик с помощью графических примитивов:



Пояснение к задаче и алгоритм решения

Рекомендуется для размещения объектов на холсте предварительно представить их координаты на бумаге, с учетом, что значение координаты  $y$  возрастает при перемещении вниз.

Порядок изображения и расположения объектов зависит только от вас.

Обратите внимание, что заливка используется только для замкнутых фигур (круга, многоугольника, овала)

```
# Подключаем графический модуль
from graph import *
```

```
# Устанавливаем ширину и высоту рабочей области окна
# Устанавливаем координаты области рисования
windowSize (800, 600)
canvasSize (700, 500)
```

```
# Устанавливаем толщину и цвет пера
penColor ("grey")
penSize (2)
```

```
# Устанавливаем цвета заливки и рисуем прямоугольники, задавая
# координаты противоположных углов
brushColor ("white")
rectangle (80, 100, 300, 150)
brushColor ("blue")
rectangle (80, 150, 300, 200)
brushColor ("red")
rectangle (80, 200, 300, 250)
```

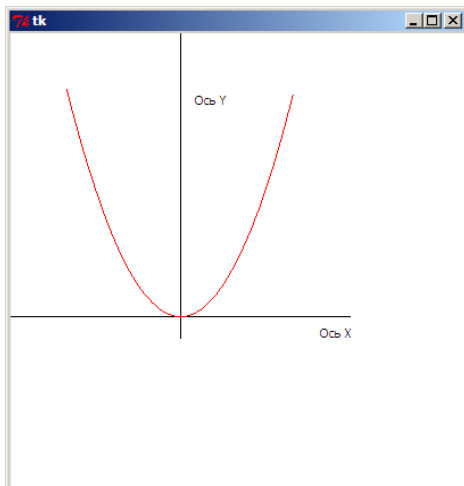
```
# Устанавливаем цвет заливки и рисуем окружность
# с центром в точке (x, y) установленного радиуса
brushColor ("yellow")
circle(550, 200, 30)
```

```
# Рисуем линию между точками с координатами
line(500, 300, 540, 220)
```

```
# Запускаем основной цикл обработки
run()
```



Пример 2.  
Построить график функции  $y=x^2$ .



Пояснение к задаче и алгоритм решения

Точка начала координат задается переменными  $x_0$  и  $y_0$  (0;0 в математике).

Диапазон изменения  $x$  - от  $x_{\min}$  до  $x_{\max}$ .

Оси координат проходят через точку  $x_0$  и  $y_0$ .

График функции строится из отрезков линий между точками. Значения точек образуют список. Коэффициент  $k$  отвечает за масштаб изображения.

```
from graph import *
#Диапазон изменения x и масштаб
x0 = 150; y0 = 250; k = 50
xmin = -2; xmax = 2
```

```
windowSize (400, 400)
```

```
#Оси координат
line (0, y0, x0+150, y0);
line (x0, 0, x0, y0+20);
label("Ось X", x0+120, y0+5)
label("Ось Y", x0+10, y0-200)
```

```
x = xmin
h = 0.02
points = []
```

```
#Формирование списка точек
while x <= xmax:
    y = x*x
    xe = x0 + k*x
    ye = y0 - k*y
    points.append ( (xe, ye) )
    x += h
```

```
#Вывод графика
penColor ("red")
polyline (points)
run()
```

**Индивидуальное задание**

Номер варианта	Задания
0	Домик $y = x^2 + 6$
1	Цветок $y = x^2 + 6x + 12$
2	Машина $y = 3x^2 + 12x$
3	Колобок $y = 14x^2 - 3$
4	Пирамидка $y = 5x^2 - 6x + 1$
5	Кукла $y = x^2 + 4x + 1$
6	Кубик $y = 9x^2 + 3x$
7	Снеговик $y = x^2 + 4x$
8	Светофор $y = 9x^2 + 16x$
9	Стол $y = 2x^2 + 6x + 5$
10	Лодка $y = 2x^2 - 4x + 1$
11	Гриб $y = 9x^2 - 3$
12	Рыбка $y = x^2 - 4x$
13	Морковка $y = 0.5x^2 - 4$
14	Бабочка $y = 2x^2 + x - 5$

### 13. ЛАБОРАТОРНАЯ РАБОТА № 11. АНИМАЦИЯ В PYTHON

#### Общие сведения

*Цель:*

Освоить технологию создания анимации в Python.

*Материалы, оборудование, программное обеспечение:*

Компьютерные классы с установленным высокоуровневым языком программирования Python.

*Условия допуска к выполнению:*

Изучить конспект по теоретической подготовке и теоретическое введение.

*Критерии положительной оценки:*

Для успешной сдачи лабораторной работы следует показать и объяснить программу, представить отчет в соответствии требованиями работы и пройти защиту.

*Планируемое время выполнения:*

Аудиторное время выполнения (под руководством преподавателя): 2 ч.

Время самостоятельной подготовки: 3 ч.

#### Теоретическое введение

Имитация движения, создаваемая серией изображений, является анимацией.

Общий алгоритм анимации:

рисование объекта,

пауза,

изменение координат объекта,

рисование объекта...

Фрагменты рисунка представляют объекты, которые потом можно переместить (перерисовывать):

obj = функция для рисования объекта.

Например:

figure = rectangle (80, 100, 300, 150)

Функции для работы с графическими объектами		
Действие	Функция	Пример
Определение объекта	obj = функция для рисования объекта	shar = circle(x, y, 30)
Функция возвращает x-координату левого верхнего угла прямоугольника	xCoord(obj)	x1=xcoord(shar)
Функция возвращает y-координату левого верхнего угла прямоугольника	yCoord(obj)	y1=ycoord(square)
Установить для объекта obj новые координаты противоположных углов прямоугольника, (x1,y1) и (x2,y2), в который вписано изображение объекта	changeCoords(obj, pos)	pos=[(10,100),(50,100)] changeCoords(linia, pos)

Функции для работы с графическими объектами		
Действие	Функция	Пример
Установить для объекта obj новый цвет контура color	changePenColor(obj, color)	changePenColor(h, "red")
Установить для объекта obj новый цвет заливки color	changeFillColor(obj, color)	changeFillColor(h, "black")
Установить для объекта obj новые свойства	changeProperty(obj, ...)	changeProperty(linia, fill="green")
Переместить левый верхний угол объекта obj в точку с координатами (x,y)	moveObjectTo(obj, x, y)	moveObjectTo(linia, 10, 10)
Переместить объект obj на вектор (dx,dy)	moveObjectBy(obj, dx, dy)	moveObjectBy(linia, 10, 10)
Удалить объект по ссылке	deleteObject(obj)	deleteObject(l)

Чтобы организовать срабатывание (вызов) функции через определенные промежутки времени, применяют «обработчик события». Событие – это изменение состояния программы или некоторое действие пользователя.

onTimer(fn, time) означает установить функцию fn, которая будет вызываться по таймеру каждые time миллисекунд.

onKey(key), onKey(fn), onKey(key, fn) означает установить функцию fn как обработчик нажатия клавиши с символьным обозначением key;

если функция не указана, обработчик нажатия этой клавиши отключается;

если не указана клавиша, устанавливается один обработчик на все клавиши;

функция fn должна принимать один параметр – блок данных о событии.

Литература:

[2] - гл. 12, с. 73-104.

*Контрольные вопросы для самопроверки:*

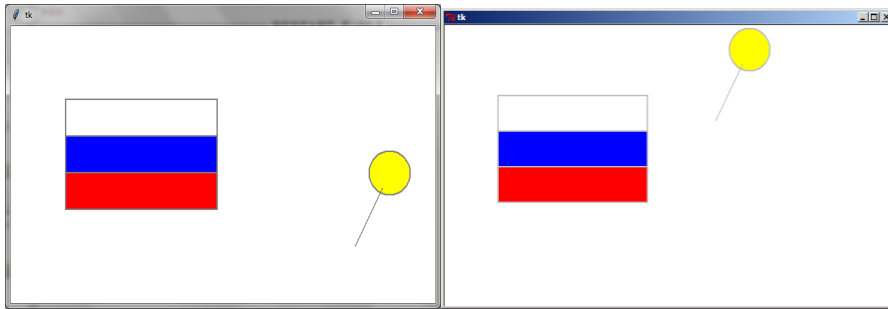
1. Что понимают под объектом при создании анимации?
2. Какие средства языка предназначены для перемещения объектов?
3. Что такое "обработчики событий области рисования"?
4. Как задать шаг изменения координат при перемещении объекта?
5. Как завершить работу программы?

### **Задание к лабораторной работе**

Написать на языке Python программу анимации для графического объекта предыдущей работы, Перемещение управлять клавишами стрелками. Траекторию и завершение движения определите по варианту.

### **Методические указания и порядок выполнения работы**

Пример 1. Организовать "полёт" шарика:



Пояснение к задаче и алгоритм решения

Рисунки определяются как объекты, с которыми работает программа.

В программе используются три функции:

- изменения координат объектов при фиксации определенных клавиш,
- перемещения объектов,
- проверки выхода объектов за границы рисунка.

В программе применяют "обработчики событий" и функции, которые будут вызываться по таймеру каждые time миллисекунд

Программа:

```

from graph import *
canvasSize (700, 500)
windowSize (800, 600)

# Функция определения шагов перемещения (dx и dy)
# при нажатии на клавиши – стрелки или клавишу пробел
# Функция закрывает окно при нажатии на клавишу ESC
def keyPressed (event):
    global dx, dy
    if event.keycode == VK_LEFT:
        dx = -5; dy = 0
    elif event.keycode == VK_RIGHT:
        dx = 5; dy = 0
    elif event.keycode == VK_UP:
        dx = 0; dy = -5
    elif event.keycode == VK_DOWN:
        dx = 0; dy = 5
    elif event.keycode == VK_SPACE:
        dx = dy = 0
    elif event.keycode == VK_ESCAPE:
        close()

# Функция перемещения объектов
def update():
    moveObjectBy(shar, dx, dy)
    moveObjectBy(linia, dx, dy)

# Функция проверки выхода фигуры за край поля
def end():

    if 0>min (coords(shar)) or 600<max(coords(shar)):
        print ('Шарик улетел')
        close()

```

```

#Основная программа
penColor("grey")
penSize(2)
brushColor("white")
rectangle(80, 100, 300, 150)
brushColor("blue")
rectangle(80, 150, 300, 200)
brushColor("red")
rectangle(80, 200, 300, 250)
# Определяем начальное положение фигур
x = 550; y = 200
# Задаем нулевые значения шагов изменения координат
dx = 0; dy = 0
brushColor("yellow")
# Определяем объекты для анимации
shar = circle(x, y, 30)
linia = line(x-50, y+100, x-10, y+20)

# Выполняем функцию как обработчик нажатия клавиш keyPressed
onKey (keyPressed)

# Определяем функции update и end,
# которые будет вызываться по таймеру каждые 50 миллисекунд
onTimer (update, 50)
onTimer (end, 50)
run()

```

### Индивидуальное задание

Номер варианта	Задания
0	Движение от верхнего правого угла поля к нижнему левому, заканчивается при достижении левого края поля
1	Движение по полю снизу вверх, заканчивается при достижении верхнего края поля
2	Движение по полю налево и направо с разной скоростью, заканчивается при достижении края поля
3	Движение по полю сверху вниз, заканчивается при достижении края поля
4	Движение только по диагонали поля, заканчивается при нажатии клавиши ESC
5	Движение по полю вверх и вниз с разной скоростью, заканчивается при достижении края поля
6	Движение в любых направлениях, заканчивается при нажатии клавиши ESC
7	Движение от левого нижнего угла поля к правому верхнему, заканчивается при достижении правого края поля
8	Движение по полю в разных направлениях, заканчивается только при достижении нижнего края поля
9	Движение от верхнего правого угла поля к нижнему левому с разной скоростью, заканчивается при нажатии клавиши ESC
10	Движение по полю сверху вниз, заканчивается при достижении края поля
11	Движение только по диагонали поля, заканчивается при нажатии клавиши ESC

Номер варианта	Задания
12	Движение по полю вверх и вниз с разной скоростью, заканчивается при достижении края поля
13	Движение в любых направлениях, заканчивается при нажатии клавиши ESC
14	Движение от левого верхнего угла поля к правому нижнему, заканчивается при достижении правого края поля

## 14. ЛАБОРАТОРНАЯ РАБОТА № 12. ВВЕДЕНИЕ В ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ. СОЗДАНИЕ КЛАССОВ. АТТРИБУТЫ И МЕТОДЫ

### Общие сведения

#### *Цель:*

Освоить основные понятия объектно-ориентированного программирования (ООП). Получить навыки создания классов с использованием атрибутов и методов.

#### *Материалы, оборудование, программное обеспечение:*

Компьютерные классы с установленным высокоуровневым языком программирования Python.

#### *Условия допуска к выполнению:*

Изучить конспект по теоретической подготовке и теоретическое введение.

#### *Критерии положительной оценки:*

Для успешной сдачи лабораторной работы следует показать и объяснить программу, представить отчет в соответствии требованиями работы и пройти защиту.

#### *Планируемое время выполнения:*

Аудиторное время выполнения (под руководством преподавателя): 6 ч.

Время самостоятельной подготовки: 3 ч.

### Теоретическое введение

К методам программирования относятся: структурное программирование и объектно-ориентированное программирование. В основе структурного программирования лежит представление программы в виде иерархической структуры блоков. Любая программа строится из базовых управляющих структур:

- последовательность
- ветвление
- цикл

Также используются подпрограммы. При этом разработка программы ведётся пошагово, методом "сверху вниз". Циклы, ветвления и функции – все это элементы структурного программирования Python.

В реальном мире люди воспринимают мир как множество объектов – предметов, животных, людей. Все объекты имеют внутреннее устройство и состояние, свойства (внешние характеристики) и поведение.

Появилась новая идея – применить в разработке программ тот подход, который использует человек в повседневной жизни.

Концепция ООП заключается в обобщении данных и функций, для ООП это атрибуты (или свойства) и методы (или функции).

Объектно-ориентированное программирование основано на концепции объектов, а не действий, и данных, а не логики.

Для того чтобы язык программирования был объектно-ориентированным, он должен иметь механизм, позволяющий работать с классами и объектами, а также реализовывать и использовать основные объектно-ориентированные принципы и концепции, а именно **наследование, абстракцию, инкапсуляцию и полиморфизм** (понятия будут рассмотрены в следующих работах).

Python соответствует принципам объектно-ориентированного программирования. В Python всё является объектами - и строки, и списки, и словари, и всё остальное.

Класс – фрагмент кода, в котором объявлены атрибуты и методы. Классы подобны чертежам: это не объекты, а схемы объектов.

Класс Студент

<b>Студент</b>
Фамилия, группа, возраст, пол, курс, направление...
Сдаёт экзамен, посещает занятия, выполняет задание...

При создании экземпляров класса получаем объект:

объект=Класс()

У объектов (экземпляров) одного и того же класса будет единообразная структура.

Объекты:

Студент 1	Студент 2
Голубева, 17 лет, 21-ВТ, 1 курс...	Яковлев, 18 лет, 20-АП, 2 курс...
Сдаёт экзамены Выполняет задания Получает повышенную стипендию	Пропускает занятия... Сдаёт не все экзамены... Представлен на отчисление..

Синтаксис объявления класса:

```
class ClasseName:
```

```
    объявление_переменных_класса
```

```
    объявление_методов_класса
```

Имена классов в Python принято начинать с прописной буквы, а имена объектов – со строчной буквы.

Пример (создание пустого класса):

```
class Empty:
```

```
    pass
```

Здесь pass – заполнитель, который можно будет заменить, как только класс получит свои атрибуты и методы.

К членам класса можно обращаться, используя синтаксис: имя\_класса.имя\_метода или имя\_класса.имя.атрибута.



К методам и атрибутам внутри класса можно обратиться, используя точечную запись с префиксом self. Параметр self метода указывает на объявленный атрибут класса.

Пример:

```
class Primer:
#объявление атрибутов класса
    группа='21-ИЭ'

#объявление метода класса
    def show (self):
        print ('Студент группы ', self.группа)

name=Primer()
name.show()
```

Студент группы 21-ИЭ

>>>

Используем предыдущую программу для получения списка студентов двух групп. Обратите внимание, что в списке параметров функции self всегда должен стоять на первом месте:

```
class Spisok:

#объявление переменных класса
    grpi='21-ИЭ'
    grvt='21-ВТ'
#объявление методов класса
    def showi (self, k):
        print ('Студент группы ', self.grpi, k)
    def showv (self,k):
        print ('Студент группы ', self.grvt, k)
```

```
fami=[' Иванов', ' Петров', ' Смирнов', ' Николаев']
famv=[' Алексеев', ' Яковлев']
name=Spisok()
print ('Группа 21-ИЭ')
for k in fami:
    name.showi(k)
print('Группа 21-ВТ')

for k in famv:
    name.showv(k)
```

Группа 21-ИЭ

Студент группы 21-ИЭ Иванов

Студент группы 21-ИЭ Петров

Студент группы 21-ИЭ Смирнов

Студент группы 21-ИЭ Николаев

Группа 21-ВТ

Студент группы 21-ВТ Алексеев

Студент группы 21-ВТ Яковлев

>>>

Выводы:

**Класс** — образец, создаваемый пользователем для объекта. Он определяет набор атрибутов, которые будут характеризовать любой объект, который создается из этого класса.

**Объект** — экземпляр класса. Это реализованная версия класса, где он проявляется в программе.

**Метод** — это особый вид функции, который определен внутри класса.

**Аргументом** метода является слово `self`, которое является ссылкой на объекты, созданные на основе этого класса. При ссылке на экземпляры (или объекты) класса, `self` всегда будет первым параметром.

*Литература:*

[3]- гл. 6, с. 21-23.

*Контрольные вопросы для самопроверки:*

1. Что такое атрибуты и методы класса?
2. В чем отличие класса от объекта?
3. Какая разница между методом и функцией?
4. Какой параметр указывается при использовании в методе атрибута класса?

### **Задание к лабораторной работе**

Написать на языке Python программу согласно своему варианту с созданием и использованием класса.

Для вывода и решения использовать методы.

Исходные данные определить вне класса.

При решении предусмотреть отсутствие данных, удовлетворяющих условию.

### **Методические указания и порядок выполнения работы**

Пример.

Исходные данные: список экзаменов и список студентов с оценками за экзамены.

Задание: создать класс, определяющий студентов, не сдавших экзамены с указанием наименования предмета, по которому студент получил неудовлетворительную оценку.

Исходные данные:

```
subject = ['Физика', 'Химия', 'Информатика']
```

```
group=[(' Арбузов', 5, 5, 5), (' Клюквин', 3, 2, 4), ('Белов',4, 2, 2), (' Елкин',4, 5, 4),...]
```

Создаем класс:

```
class Session():
```

Класс содержит:

атрибут `ball` со значением "2"

метод `verify`

В методе `verify` для каждого элемента исходного списка проверим наличие хотя бы одной неудовлетворительной оценки (атрибут `ball`). В этом случае определим её индекс, по которому выведем название предмета.

Например, для элемента:

(' Клюквин', 3, 2, 4) индекс равен 1, т.е. 'Химия'

Создается объект класса:

```
itog=Session()
```

Каждый элемент исходного списка данных обрабатывает метод `verify` :

```
for fam in group:
```

```
    itog.verify(fam)
```

Переменная n предназначена для определения в программе наличия (или отсутствия) неуспевающего студента.

Программа:

```
class Session():
```

```
#объявление переменных класса
```

```
    ball = 2
```

```
#объявление_методов_класса
```

```
    def verify (self, fam):
```

```
        global n
```

```
        if self.ball in fam:
```

```
            n +=1
```

```
            print ('Студент ', fam[0], ' - двоечник')
```

```
            print ('    Предмет: ')
```

```
            for i in range (1,4):
```

```
                if fam[i]==self.ball:
```

```
                    print ('        ', subject[i-1])
```

```
subject = ['Физика','Химия','Информатика']
```

```
group=[(' Арбузов', 5, 5, 5), (' Клюквин', 3, 2, 4), ('Белов', 4, 2, 2), (' Елкин', 4, 5, 4),  
        ('Носов', 4, 3, 3)]
```

```
n=0
```

```
itog=Session()
```

```
for fam in group:
```

```
    itog.verify(fam)
```

```
if n == 0:
```

```
    print ('В группе все студенты сдали экзамены')
```

```
Результат:
```

```
        Студент Клюквин - двоечник
```

```
        Предмет:
```

```
        Химия
```

```
        Студент Белов - двоечник
```

```
        Предмет:
```

```
        Химия
```

```
        Информатика
```

```
>>>
```

```
Или (при других данных):
```

```
        В группе все студенты сдали экзамены
```

```
>>>
```

### Индивидуальное задание

Номер варианта	Исходный список (не менее шести элементов)	Задача
0	Список студентов состоит из фамилий, количества задолженностей (или их отсутствия), формы обучения	Вывести списки задолжников по каждой форме обучения
1	Список автомобилей содержит марку и год выпуска	Вывести списки автомобилей, выпущенных до и после указанного года

Номер варианта	Исходный список (не менее шести элементов)	Задача
2	Список студентов состоит из фамилий и наименования группы	Вывести списки студентов, обучающихся по определенному направлению и всех остальных
3	Список групп содержит наименование группы и количества студентов в группе	Вывести список групп заданного года поступления
4	Список студентов состоит из фамилий, количества задолженностей, даты ликвидации задолженностей	Вывести список студентов, которые должны ликвидировать задолженности к указанной дате
5	Список состоит из наименований групп и количества студентов в группе	Вывести список групп и количество студентов определенного курса
6	Список состоит из наименований направлений, количества студентов по каждому курсу	Вывести список направлений с общим числом студентов более указанного значения
7	Список спортсменов состоит из фамилий и занятых мест в соревновании	Вывести списки спортсменов, занявших первые три места и остальных
8	Список студентов состоит из фамилий, номера семестра, количества сданных предметов	Вывести списки студентов, сдавших указанную сессию, и студентов - задолжников
9	Список студентов состоит из фамилий и оценок за экзамены	Вывести список отличников, список двоечников, список студентов, сдавших экзамены
10	Список спортсменов состоит из фамилий и квалификации	Вывести списки спортсменов определенной квалификации и остальных
11	Список студентов состоит из фамилий и наименования группы	Вывести списки студентов определенного года поступления и остальных
12	Список городов содержит наименования города и страны	Вывести списки городов двух указанных стран
13	Список студентов состоит из фамилий и количества пропущенных часов занятий	Вывести списки студентов, не пропускающих занятия, и остальных
14	Список студентов состоит из фамилий и года рождения	Вывести список студентов, родившихся в указанном году и список студентов, которым не исполнилось 18 лет

## **15. ЛАБОРАТОРНАЯ РАБОТА № 13. ПРИНЦИПЫ ОБЪЕКТНО-ОРИЕНТИРОВАННОГО ПРОГРАММИРОВАНИЯ. НАСЛЕДОВАНИЕ. ИНКАПСУЛЯЦИЯ**

### **Общие сведения**

#### *Цель:*

Овладеть одним из принципов объектно-ориентированного программирования: наследованием. Получить навыки использования классов - наследников. Освоить применение инкапсуляции и конструктора как способов создания объектов.

*Материалы, оборудование, программное обеспечение:*  
Компьютерные классы с установленным высокоуровневым языком программирования Python.

*Условия допуска к выполнению:*

Изучить конспект по теоретической подготовке и теоретическое введение.

*Критерии положительной оценки:*

Для успешной сдачи лабораторной работы следует показать и объяснить программу, представить отчет в соответствии требованиями работы и пройти защиту.

*Планируемое время выполнения:*

Аудиторное время выполнения (под руководством преподавателя): 6 ч.

Время самостоятельной подготовки: 3 ч.

## Теоретическое введение

Один из принципов объектно-ориентированного программирования: наследование. Наследование позволяет акцентировать иерархические отношения между классами и объектами. Например, "время года" является обобщением "сезон весна", "домашнее животное" – "кошка".

Наследование очень полезно с точки зрения повторного использования кода.

Основная идея наследования в объектно-ориентированном программировании заключается в том, что **класс может наследовать характеристики другого класса**. Класс, который наследует другой класс, называется **дочерним классом** или производным классом, и класс, который дает наследие, называется **родительским**, или основным.

Наследование в объектно-ориентированном программировании очень похоже на наследование в реальной жизни, где ребенок наследует те или иные характеристики его родителей в дополнение к его собственным характеристикам.

Классы наследники объявляются так, как и родительские классы. Но наследуемый класс указывается после имени основного класса:

### Пример 1.

```
class Base():  
    pass  
class Subclass(Base):  
    pass
```

...

Наследование атрибутов можно рассмотреть на примере:

```
class Date():  
    def day(self):  
        return '28.05.2022'  
class Time(Date):  
    def clock(self):  
        return '14:00'
```

```
dt=Date()  
print ('Дата класса Date:', dt.day())  
tm=Time()  
print ('Время класса Time:', tm.clock())
```

```
print ('Дата класса Time (учитывая наследование метода класса Date):', tm.day())
```

Дата класса Date: 28.05.2022

Время класса Time: 14:00

Дата класса Time (учитывая наследование метода класса Date): 28.05.2022

>>>

Создан класс с именем Date, далее создан еще один класс Time, в качестве аргумента – класс Date. Это позволит получить доступ ко всем данным и атрибутам класса Date в классе Time. Поэтому возможно использовать метод day из объекта класса Time.

Скрытие внутреннего устройства объектов называют **инкапсуляцией** ("помещение в капсулу"). Как правило, в объектно-ориентированном программировании один класс не должен иметь прямого доступа к данным другого класса. Вместо этого доступ должен контролироваться через методы класса.

Инкапсуляция - объединение данных и методов работы с этими данными в один объект, позволяя скрыть детали реализации объекта от пользователя. Инкапсуляция — ограничение доступа к составляющим объект компонентам (методам и переменным). Инкапсуляция делает некоторые из компонент доступными только внутри класса.

Инкапсуляция в Python работает лишь на уровне соглашения между программистами о том, какие атрибуты являются общедоступными, а какие — внутренними.

Одинокое подчеркивание в начале имени атрибута говорит о том, что переменная или метод не предназначен для использования вне методов класса, однако атрибут доступен по этому имени.

```
class Keep_away1:
    def _hide(self):
        print("_hide(self) - приватный метод припрятывать!")
w = Keep_away1()
w._hide()
```

Два подчеркивания в начале имени атрибута даёт большую защиту: атрибут становится недоступным по этому имени.

```
class Keep_away2:
    def __secrete(self):
        print("__secrete(self) - приватный метод прятать")
w = Keep_away2()
```

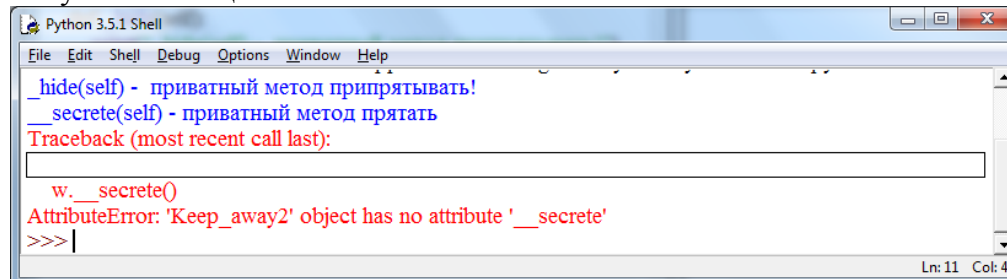
Атрибут всё равно остаётся доступным под именем: ИмяКласса\_\_ИмяАтрибута:

```
w._Keep_away2__secrete()
```

При выполнении инструкции:

```
w.__secrete()
```

получаем сообщение об ошибке:



**Конструктор** — уникальный метод класса, который называется `__init__`.

Первый параметр конструктора во всех случаях `self` (ключевое слово, которое ссылается на сам класс).

Конструктор нужен для создания объекта.

Конструктор передает значения аргументов свойствам создаваемого объекта.

В одном классе всегда только один конструктор.

Если класс определяется не конструктором, Python предположит, что он наследует конструктор родительского класса.

Пример вычисления площади треугольника.

Внутри конструктора устанавливаются два атрибута - `base` и `height` (основание и высота треугольника). Через эти параметры в конструктор передаются значения элементов создаваемого объекта.

Метод конструктора инициализируется автоматически, поэтому его не нужно явно вызывать. Достаточно при создании экземпляра класса передать аргументы в круглых скобках, следующих за именем класса.

Метод `value_area` предназначен для вычисления площади треугольника.

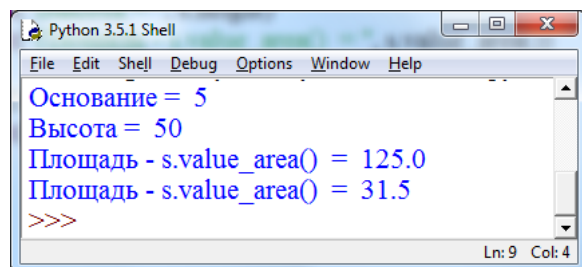
В программе можно использовать и значения параметров `base` и `height`:

`s.base` и `s.height`

```
class Triangle:
    # Конструктор - способ создания объекта
    def __init__(self, base, height):
        self.base= base
        self.height = height

    def value_area (self):
        return self.base * self.height/2

s = Triangle (5,50)
print("Основание = ", s.base)
print("Высота = ", s.height)
print("Площадь - s.value_area() = ", s.value_area())
s = Triangle (9,7)
print("Площадь - s.value_area() = ", s.value_area())
```



Конструктор позволяет задавать значения по умолчанию.

Пример:

```
class Favorite:
    # Параметры возраста и
    # вида домашних животных
    # имеют значение по умолчанию
    def __init__(self, name, age=1, animal= "Собака"):
        self.name = name
        self.age = age
        self.animal= animal

    def showInfo(self):
        print("Кличка: ", self.name)
        print("Возраст: ", self.age)
        print("Домашний питомец: ", self.animal)
```

```

print()

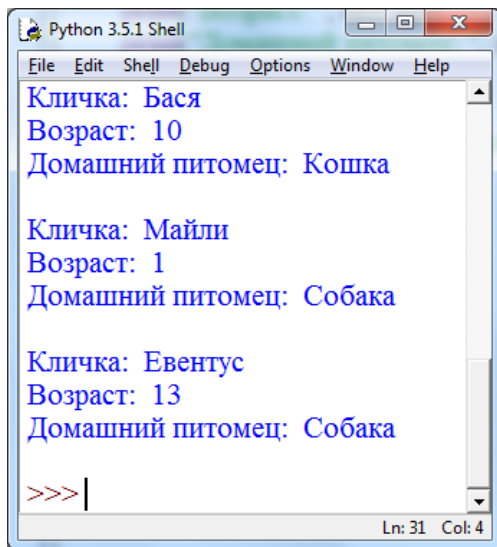
# Создание объектов Favorite

basya = Favorite ("Бася", 10, "Кошка")
basya.showInfo()

# По умолчанию возраст и вид
miley = Favorite ("Майли" )
miley.showInfo()

# По умолчанию вид
eventus = Favorite ("Евентус", 13)
eventus.showInfo()

```



```

Python 3.5.1 Shell
File Edit Shell Debug Options Window Help
Кличка: Бася
Возраст: 10
Домашний питомец: Кошка

Кличка: Майли
Возраст: 1
Домашний питомец: Собака

Кличка: Евентус
Возраст: 13
Домашний питомец: Собака

>>> |
Ln: 31 Col: 4

```

*Литература:*

[3]- гл. 6, с. 21-23.

*Контрольные вопросы для самопроверки:*

1. Что такое атрибуты и методы класса?
2. В чем отличие класса от объекта?
3. Какая разница между методом и функцией?
4. Какой параметр указывается при использовании в методе атрибута класса?

### **Задание к лабораторной работе**

Напишите на языке Python программу по варианту своего задания из таблицы. Самостоятельно задайте параметры метода для определения класса – наследника. В основном классе используйте конструктор.

При решении предусмотреть отсутствие данных, удовлетворяющих условию.

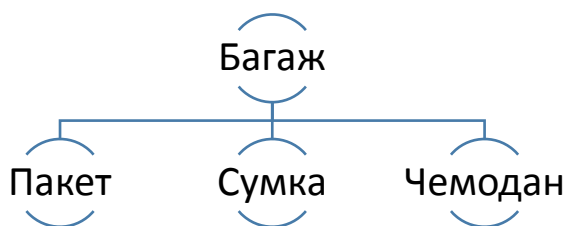
### **Методические указания и порядок выполнения работы**

Пример.

Описать класс Багаж с полем Название и методом, при вызове которого определяется назначение класса, а также классы - наследники: пакет, сумка, чемодан и т. д.



В качестве метода для определения класса – наследника используем выбор возможного веса. Пример наследования:



Создаем класс Luggage, атрибутами которого являются имя, минимальный вес, максимальный вес. Конструктор класса используется для создания объекта и присвоения значения атрибутам, метод проверяет допустимый вес, возвращая истину или ложь

```
class Luggage:
    name='Багаж'
    min_w=0
    max_w=0
    def __init__(self, weight):
        self.weight = weight

    def check_w (self):
        if self.min_w <= self.weight < self. max_w:
            return True
        return False
```

#Создаем классы наследники:

```
class Package (Luggage):
    name='Пакет'
    min_w=0
    max_w=3
```

```
class Buggage_bag (Luggage):
    name='Багажная сумка'
    min_w=3.1
    max_w=10
```

```
class Suitcase (Luggage):
    name='Чемодан'
    min_w=10.1
    max_w=25
```

#Основная программа вводит вес багажа, определяет его допустимость.

```
weigh = float (input( 'Введите вес багажа, который Вы берёте в самолет (в кг): '))
if weigh<=0:
    print ('Вы ввели не положительное число!')
else:

    your_weigh = Package (weigh)
    if your_weigh.check_w():
        print ('Достаточно взять ', your_weigh.name)
    else:
```

```

your_weigh = Baggage_bag (weigh)
if your_weigh.check_w():
    print ('Вам подойдет ', your_weigh.name)
else:
    your_weigh = Suitcase (weigh)
    if your_weigh.check_w():
        print ('Ваш багаж поместится только в ',your_weigh.name)
    else:
        print ('С таким весом в самолет не пустят!')

```

Результат:

```

Python 3.5.1 Shell
File Edit Shell Debug Options Window Help
Python 3.5.1 (v3.5.1:37a07cee5969, Dec 6 2015, 01:38:48) [MSC v.1900 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: E:/flight.py =====
Введите вес багажа, который Вы берёте в самолет (в кг): 2.5
Достаточно взять Пакет
>>>
===== RESTART: E:/flight.py =====
Введите вес багажа, который Вы берёте в самолет (в кг): 21
Ваш багаж поместится только в Чемодан
>>>
===== RESTART: E:/flight.py =====
Введите вес багажа, который Вы берёте в самолет (в кг): 9
Вам подойдет Багажная сумка
>>>
===== RESTART: E:/flight.py =====
Введите вес багажа, который Вы берёте в самолет (в кг): 46
С таким весом в самолет не пустят!
>>>
===== RESTART: E:/flight.py =====
Введите вес багажа, который Вы берёте в самолет (в кг): 0
Вы ввели не положительное число!
>>>
Ln: 20 Col: 52

```

### Индивидуальное задание

Номер варианта	Задание
0.	Описать класс Автомобиль с полем Название и методом, при вызове которого определяется страна-производитель, а также классы - наследники: Германия, Франция, Япония и т. д.
1.	Описать класс Машина с полем Название и методом, при вызове которого определяется назначение данной машины, а также классы – наследники: такси, автобус, поезд и т. д.
2.	Описать класс Спортсмен с полем Название и методом, при вызове которого определяется квалификация спортсмена, а также классы – наследники: кандидат в мастера спорта, мастер спорта и т. д.
3.	Описать класс Школа с полем Название и методом, при вызове которого определяется классы – наследники: начальные классы, выпускные классы и т. д.
4.	Описать класс Страна с полем Название и методом, при вызове которого определяется наименование государства, а также классы – наследники: Россия, Испания, Литва и т.д.

Номер варианта	Задание
5.	Описать класс Времена года с полем Название и методом, при вызове которого определяется сезоны года, а также классы – наследники: лето, весна и т. д.
6.	Описать класс Семья с полем Название и методом, при вызове которого определяется положение в семье, а также классы – наследники: сын, внучка и т. д.
7.	Описать класс Планета с полем Название и методом, при вызове которого определяется наименование планеты, а также классы – наследники: Венера, Меркурий, Марс
8.	Описать класс Сутки с полем Название и методом, при вызове которого определяется время суток, а также классы – наследники: утро, вечер и т.д.
9.	Описать класс Фигура с полем Название и методом, при вызове которого определяется вид фигуры, и классы – наследники: круг, квадрат, прямоугольник и т. д.
10.	Описать класс Студент с полем Название и методом, при вызове которого определяется результаты сдачи сессии студентом, а также классы – наследники: отличник, хорошист и т. д.
11.	Описать класс Жильё с полем Название и методом, при вызове которого определяется вид жилья, и классы – наследники: квартира, дом, и т. д.
12.	Описать класс Населенный пункт с полем Название и методом, при вызове которого определяется вид проживания, а также классы – наследники: город, поселок городского типа, деревня и т. д.
13.	Описать класс Движение с полем Название и методом, при вызове которого определяется вид движения, а также классы – наследники: полет, ходьба и т. д.
14.	Описать класс Температура с полем Название и методом, при вызове которого определяется ощущение, а также классы – наследники: жара, холод, мороз и т. д.

## 16. ЛАБОРАТОРНАЯ РАБОТА № 14. СОЗДАНИЕ ГРАФИЧЕСКОГО ИНТЕРФЕЙСА ПОЛЬЗОВАТЕЛЯ (GUI) СРЕДСТВАМИ БИБЛИОТЕКИ TKINTER

### Общие сведения

#### *Цель:*

Научиться создавать элементы с учетом конфигурации их свойств, определения событий и обработчиков событий в программе.

#### *Материалы, оборудование, программное обеспечение:*

Компьютерные классы с установленным высокоуровневым языком программирования Python.

#### *Условия допуска к выполнению:*

Изучить конспект по теоретической подготовке и теоретическое введение.

#### *Критерии положительной оценки:*

Для успешной сдачи лабораторной работы следует показать и объяснить программу, представить отчет в соответствии требованиями работы и пройти защиту.

*Планируемое время выполнения:*

Аудиторное время выполнения (под руководством преподавателя): 8 ч.

Время самостоятельной подготовки: 3 ч.

## Теоретическое введение

Tkinter – это пакет для Python, предназначенный для работы с библиотекой Tk. Библиотека Tk содержит компоненты графического интерфейса пользователя (graphical user interface – GUI), написанные на языке программирования Tcl.

Под графическим интерфейсом пользователя (GUI) подразумеваются все те окна, кнопки, текстовые поля для ввода, скроллеры, списки, радиокнопки, флажки и др., которые вы видите на экране, открывая то или иное приложение. Через них вы взаимодействуете с программой и управляете ею. Все эти элементы интерфейса вместе называются виджетами (widgets).

В настоящее время почти все приложения, которые создаются для конечного пользователя, имеют GUI.

Чтобы написать GUI-программу, надо выполнить приблизительно следующее:

Создать главное окно.

Создать виджеты и выполнить конфигурацию их свойств (опций).

Определить события, т. е. то, на что будет реагировать программа.

Определить обработчики событий, т. е. то, как будет реагировать программа.

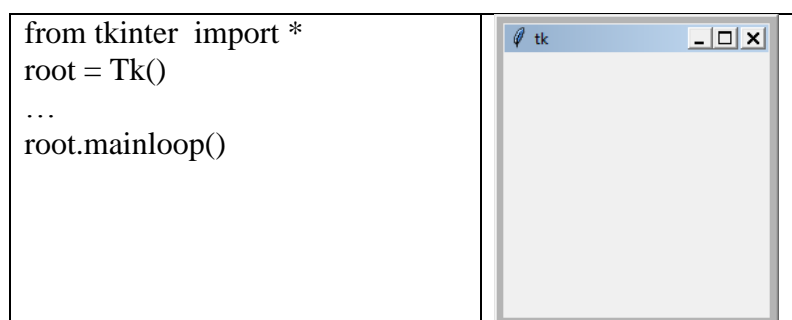
Расположить виджеты в главном окне.

Запустить цикл обработки событий.

Tk является базовым классом любого tkinter приложения. При создании объекта этого класса запускается интерпретатор tcl/tk и создаётся базовое окно приложения.

Tkinter является событийно-ориентированной библиотекой. В приложениях такого типа имеется главный цикл обработки событий. В tkinter такой цикл запускается методом **mainloop**. Для явного выхода из интерпретатора и завершения цикла обработки событий используется метод **quit**.

Таким образом, минимальное приложение на tkinter будет таким:



Виджеты создаются вызовом конструктора соответствующего класса.

Родительский виджет можно не указывать, в таком случае будет использовано главное окно приложения.

Далее следуют именованные аргументы, конфигурирующие виджет.


Это может быть используемый шрифт (`font=...`), цвет виджета (`bg=...`), команда, выполняющаяся при активации виджета (`command=...`) и т.д.

Все виджеты в tkinter обладают некоторыми общими свойствами. В таблицах 1 и 2 представлены основные сведения о виджетах и их аргументах.

В примерах представлена часть программы без создания базового окна и цикла обработки событий.

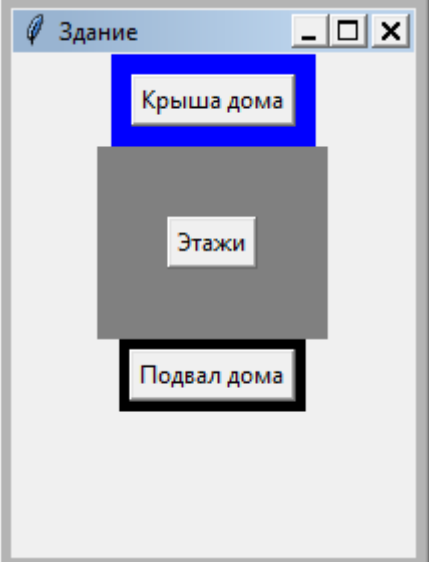
Виджет Button - самая обыкновенная "кнопка", которая используется в тысячах программ.

Пример:

<pre>root=Tk() button1=Button(root, text='КГТУ ИЦТ', width=25,                 height=5, bg='black',fg='red', font='arial 16') button1.pack()</pre>	
-----------------------------------------------------------------------------------------------------------------------------------------------------	------------------------------------------------------------------------------------

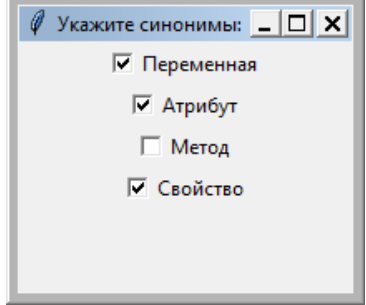
Виджет Frame предназначен для организации виджетов внутри окна.

Пример:

<pre>root=Tk() root.title ('Здание') root.geometry("200x250") рамка1=Frame(root, bg='blue', bd=10) рамка2=Frame(root, bg='grey', bd=35) рамка3=Frame(root, bg='black', bd=5) button1=Button(рамка1, text= 'Крыша дома') button2=Button(рамка2, text= 'Этажи') button3=Button(рамка3, text= 'Подвал дома') рамка1.pack() рамка2.pack() рамка3.pack() button1.pack() button2.pack() button3.pack()</pre>	
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	-------------------------------------------------------------------------------------

Виджет Checkbutton позволяет отметить "галочкой" определенный пункт или нескольких пунктов.

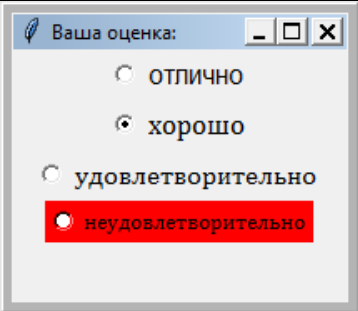
Пример:

<pre>root=Tk() var1=IntVar() var2=IntVar() var3=IntVar() var4=IntVar() root.title ('Укажите синонимы: ') root.geometry("200x150")  check1=Checkbutton(root,text= 'Переменная',variable=var1,                    onvalue=1, offvalue=0) check2=Checkbutton(root,text= 'Атрибут', variable=var2,                    onvalue=1, offvalue=0) check3=Checkbutton(root,text= 'Метод',variable=var3,                    onvalue=1, offvalue=0) check4=Checkbutton(root,text= 'Свойство', variable=var4,</pre>	
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	---------------------------------------------------------------------------------------

onvalue=1, offvalue=0)	
<pre> check1.pack() check2.pack() check3.pack() check4.pack() </pre>	

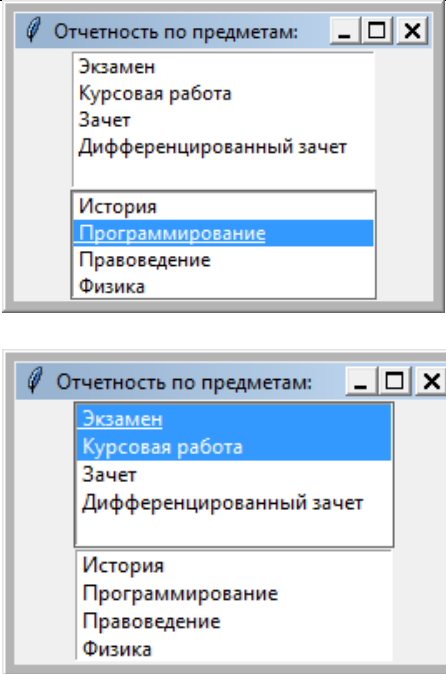
В виджете Radiobutton пользователь может выбрать лишь один из пунктов. В этом виджете используется уже одна переменная. В зависимости от того, какой пункт выбран, она меняет своё значение.

Пример:

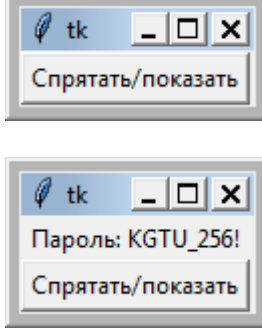
<pre> root=Tk() var=IntVar() root.title('Ваша оценка: ') root.geometry("200x150")  rbutton1=Radiobutton(root,text='отлично', font='cambria14',                       variable=var,value=1) rbutton2=Radiobutton(root,text='хорошо', font='cambria 13',                       variable=var,value=2) rbutton3=Radiobutton(root,text='удовлетворительно',                       font='cambria 12', variable=var,value=3) rbutton4=Radiobutton(root,text='неудовлетворительно',                       bg='red', font='cambria 10', variable=var, value=4)  rbutton1.pack() rbutton2.pack() rbutton3.pack() rbutton4.pack() </pre>	
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	-------------------------------------------------------------------------------------

Виджет Listbox представляет собой список, из элементов которого пользователь может выбирать один или несколько пунктов. Имеет дополнительное свойство selectmode, которое, при значении SINGLE, позволяет пользователю выбрать только один элемент списка, а при значении EXTENDED - любое количество.

Пример:

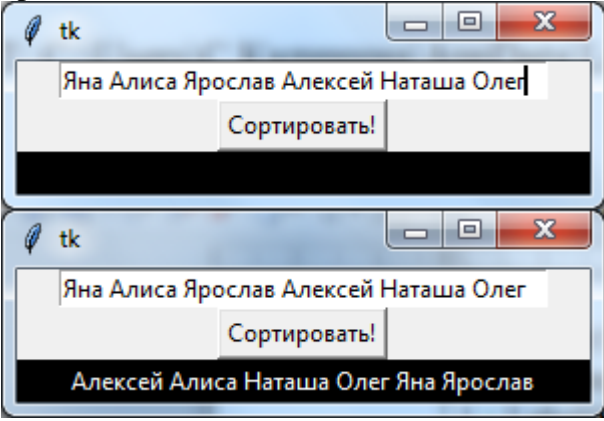
<pre> root=Tk() listbox1=Listbox(root,height=5,width=30,                  selectmode=EXTENDED) listbox2=Listbox(root,height=5,width=30,                  selectmode=SINGLE) root.title ('Отчетность по предметам: ') root.geometry("200x150") list1=[ " Экзамен", " Курсовая работа", " Зачет",         " Дифференцированный зачет"]  list2=[ " История", " Программирование",         " Правоведение", " Физика"]  for i in list1:     listbox1.insert(END,i)  for i in list2:     listbox2.insert(END,i)  listbox1.pack() listbox2.pack() </pre>	
----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	------------------------------------------------------------------------------------

Если необходимо только на время спрятать какой-либо виджет, то можно пользоваться упаковщиком grid и методом grid\_remove,  
Пример:

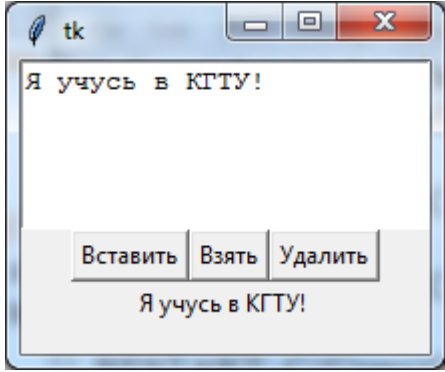
<pre> def hide_password():     if label.winfo_viewable():         label.grid_remove()     else:         label.grid() root=Tk()  label = Label(text='Пароль: KGTU_256!') label.grid() button = Button(command=hide_password, text="Спрятать/показать") button.grid() </pre>	
----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	---------------------------------------------------------------------------------------

Пример 1. В текстовом поле окна вводится список элементов, требуется создать кнопку для сортировки списка.

Версия 1	Версия 2 с использованием ОПП
<pre> root = Tk()  e = Entry(width=40) b = Button(text="Сортировать!") l = Label(bg='black', fg='white', width=40) </pre>	<pre> class Block:     def __init__(self, master):         self.e = Entry(master, width=40)         self.b = Button(master,             text=" Сортировать!")         self.l = Label(master, bg='black', </pre>

<pre> def strToSortlist(event):     s = e.get()     s = s.split()     s.sort()     l['text'] = ' '.join(s)  b.bind('&lt;Button-1&gt;', strToSortlist)  e.pack() b.pack() l.pack() </pre> 	<pre> fg='white', width=40) self.e.pack() self.b.pack() self.l.pack() def setFunc(self, func):     self.b['command'] = eval('self.' + func) def strToSortlist(self):     s = self.e.get()     s = s.split()     s.sort()     self.l['text'] = ' '.join(s) def strReverse(self):     s = self.e.get()     s = s.split()     s.reverse()     self.l['text'] = ' '.join(s)  root = Tk() #Сортировка по возрастанию first_block = Block(root) first_block.setFunc('strToSortlist') #Сортировка по убыванию second_block = Block(root) second_block.setFunc('strReverse') </pre>
----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

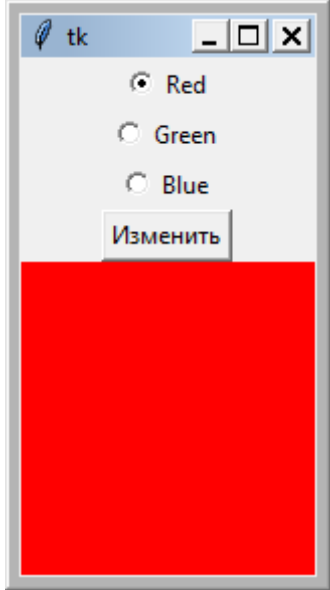
Пример 2. Написать программу для нижеприведенного вида окна. При нажатии на кнопки текст надписи вставляется в текстовое поле или удаляется из него.

<pre> def insertText():     s = "Я учусь в КГТУ! "     text.insert(1.0, s)  def getText():     s = text.get(1.0, END)     label['text'] = s  def deleteText():     text.delete(1.0, END)  root = Tk()  text = Text(width=25, height=5) text.pack()  frame = Frame() frame.pack()  b_insert = Button(frame, text="Вставить", command=insertText) b_insert.pack(side=LEFT)  b_get = Button(frame, text="Взять", </pre>	
----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------



<pre> command=getText) b_get.pack(side=LEFT)  b_delete = Button(frame, text="Удалить", command=deleteText) b_delete.pack(side=LEFT)  label = Label() label.pack() </pre>	
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------	--

Пример 3. В зависимости от выбора цвета с помощью флажка цвет метки (без текста) меняется на выбранный.

<pre> def change():     if var.get() == 0:         label['bg'] = 'red'     elif var.get() == 1:         label['bg'] = 'green'     elif var.get() == 2:         label['bg'] = 'blue'  root = Tk()  var = IntVar() var.set(0) red = Radiobutton(text="Red", variable=var, value=0) green = Radiobutton(text="Green", variable=var, value=1) blue = Radiobutton(text="Blue", variable=var, value=2) button = Button(text="Изменить", command=change) label = Label(width=20, height=10) red.pack() green.pack() blue.pack() button.pack() label.pack() </pre>	
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------

#### Литература:

[2] - гл. 12, с. 73-103.

[3] – гл. 23, с. 77-85.

#### Контрольные вопросы для самопроверки:

1. Что понимают под виджетами?
2. Что образует главный цикл обработки событий в tkinter?
3. Как организовано размещение виджетов?
4. Какие свойства общие для всех виджетов?
5. Назначение виджетов Button, Label, Entry, Text?
6. Назначение виджетов Listbox, Frame, Radiobutton, Scrollbar?

#### Задание к лабораторной работе

Ваша программа на языке Python должна:

1. Создать главное окно.
2. Создать виджеты по варианту и выполнить конфигурацию их свойств (опций).  
**Дизайн формы придумайте самостоятельно**
3. Определить события, то есть то, на что будет реагировать программа.
4. Определить обработчики событий, то есть то, как будет реагировать программа.
5. Расположить виджеты в главном окне.
6. Запустить цикл обработки событий.

(Последовательность не обязательно такая, но первый и последний пункты всегда остаются на своих местах).

### Методические указания и порядок выполнения работы

Воспользуйтесь примерами, приведенными в теоретической части работы и данными таблиц: виджеты, опции виджетов и назначения.

Таблица 1. Виджеты и назначения

Виджет		Назначение
Button	Кнопка	Кнопка с указанием её свойств (начинать нужно с указания окна)
Label	Метка	Виджет предназначен для отображения какой-либо надписи без возможности редактирования
Entry	Запись	Виджет для ввода одной строки текста
Text	Текст	Виджет для ввода текста
Listbox	Список	Виджет списка, из элементов которого выбирают один или несколько пунктов.
Frame	Рамка	Виджет предназначен для организации виджетов внутри виджета
Checkbutton	Кнопка проверки	Виджет, который позволяет отметить определенный пункт в окне. При использовании нескольких пунктов нужно каждому присвоить свою переменную
Radiobutton	Переключатель	Виджет выполняет функцию, схожую с функцией виджета Checkbutton. Разница в том, что в виджете Radiobutton пользователь может выбрать лишь один из пунктов
Scrollbar	Полоса прокрутки	Этот виджет даёт возможность пользователю "прокрутить" другой виджет (например, текстовое поле) Использование этого виджета достаточно нетривиально. Необходимо сделать две привязки: command полосы прокрутки привязываем к методу xview/yview виджета, а xscrollcommand/yscrollcommand виджета привязываем к методу set полосы прокрутки
pack(), place(), grid()	Упаковщики	Специальный механизм, который размещает (упаковывает) виджеты на окне. В одном виджете можно использовать только один тип упаковки
command		Активацию виджета на действие пользователя (нажатие кнопки)
bind()		Метод привязывает событие к какому-либо действию (нажатие кнопки мыши, нажатие клавиши на клавиатуре и т.д.) bind принимает три аргумента: <ul style="list-style-type: none"> <li>• название события</li> </ul>

Виджет		Назначение
		<ul style="list-style-type: none"> <li>• функцию, которая будет вызвана при наступлении события</li> <li>• третий аргумент (необязательный) - строка "+" - означает, что эта привязка добавляется к уже существующим. Если третий аргумент опущен или равен пустой строке - привязка замещает все другие привязки данного события к виджету</li> </ul>

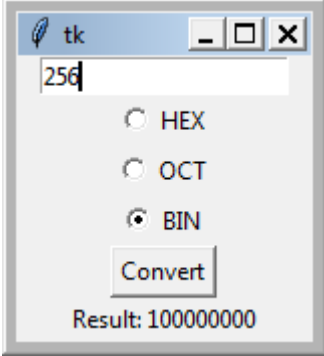
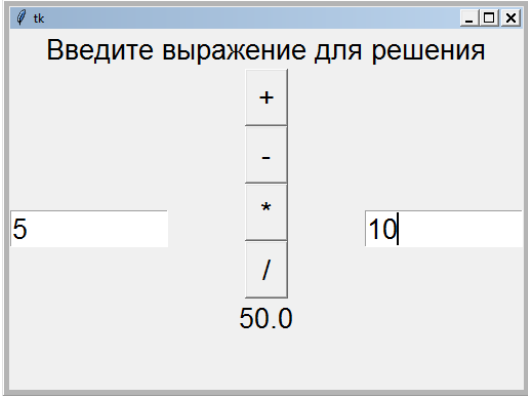
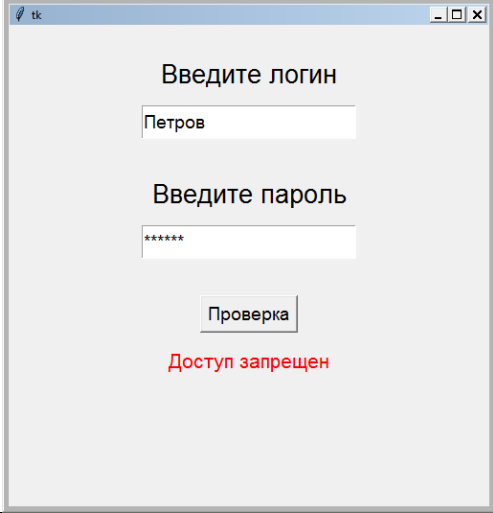
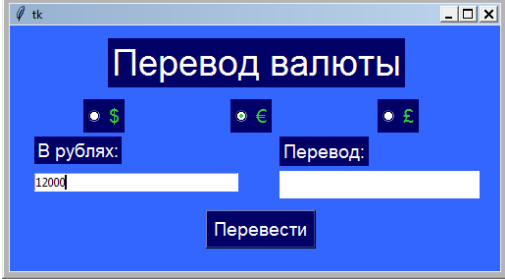
Таблица 2. Опции виджетов и назначения

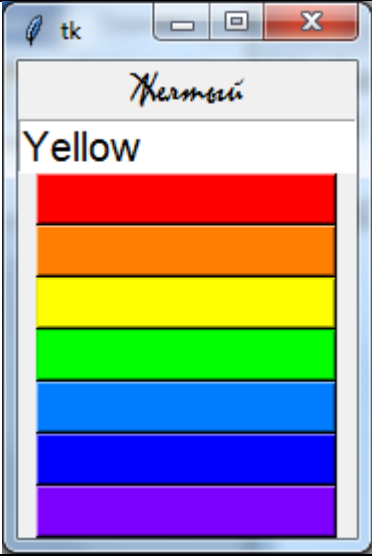
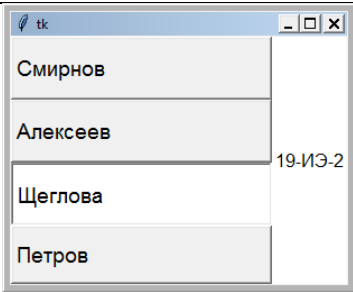
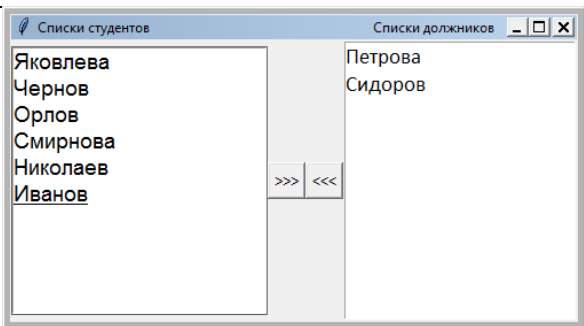
Виджет	Аргументы	Значения
Button label	text	Текст на кнопке
	width, height	Ширина и длина кнопки
	bg	Цвет кнопки (сокращенно от background)
	font	Шрифт и его размер ("arial 14")
Entry	bd	Регулировка ширины границы (сокращенно от borderwidth)
	width	Задаёт длину элемента в знаках
	show	Задаёт отображаемый символ
Text	wrap	Перенос текста (для переноса по словам, нужно использовать значение WORD)
Listbox	selectmode ( SINGLE, EXTENDED)	SINGLE позволяет пользователю выбрать только один элемент списка, при значении EXTENDED - любое количество
Frame	bd	Свойство отвечает за толщину края рамки
Checkbutton	IntVar() - специальный класс библиотеки для работы с целыми числами variable	Свойство, отвечающее за прикрепление к виджету переменной
	onvalue	Свойство, которое присваивают прикрепленной к виджету переменной значение, которое зависит от состояния при выбранном пункте
	offvalue	Свойство, которое присваивают прикрепленной к виджету переменной значение, которое зависит от состояния при невыбранном пункте
Scrollbar	command xview/yview	Привязки: command полосы прокрутки привязываются к методу xview/yview виджета
	command xscrollcommand/ yscrollcommand	Привязки: command полосы прокрутки привязываются к xscrollcommand/yscrollcommand к методу set полосы прокрутки
Pack()	side ("left"/"right"/"top"/"bottom")	К какой стороне должен примыкать размещаемый виджет

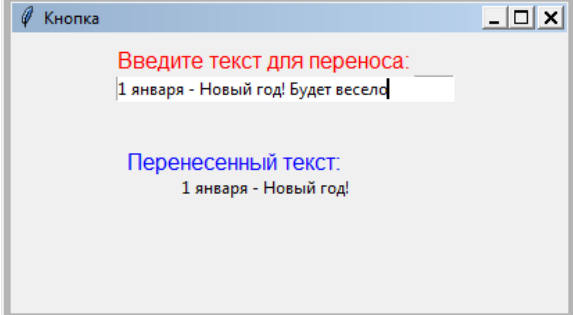
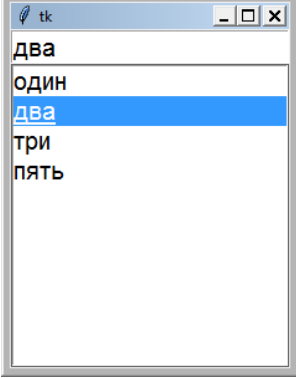
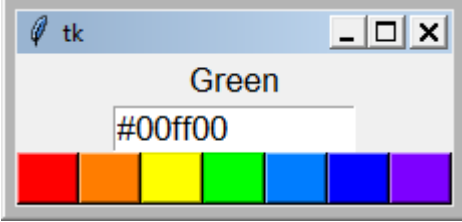
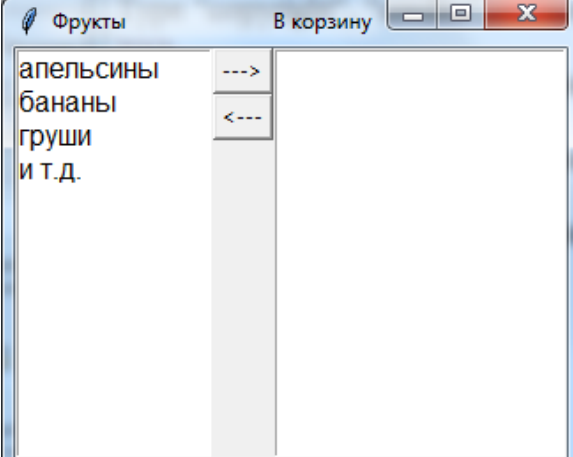
Виджет	Аргументы	Значения
	fill (None/"x"/"y"/"both")	Определяет необходимость расширения пространства виджету
	expand (True/False)	Определяет необходимость расширения самого виджета, чтобы он занял всё предоставляемое ему пространство
	in	Явное указание: в какой родительский виджет должен быть помещён
Grid()	row column	Номер строки и номер столбца, в который помещается виджет
	rowspan columnspan	Сколько строк и сколько столбцов занимает виджет
	padx / pady	Размер внешней границы (бордюра) по горизонтали и вертикали, расширяется свободное пространство
	ipadx / ipady	Размер внутренней границы (бордюра) по горизонтали и вертикали, расширяется помещаемый виджет
	sticky ("n", "s", "e", "w" или их комбинация)	Указывает: к какой границе "приклеивать" виджет. Позволяет расширять виджет в указанном направлении. Границы названы в соответствии со сторонами света. "n" (север) - верхняя граница, "s" (юг) - нижняя, "w" (запад) - левая, "e" (восток) - правая
Place()		Простой упаковщик, позволяющий размещать виджет в фиксированном месте с фиксированным размером. При использовании этого упаковщика, необходимо указывать координаты каждого виджета
	anchor ("n", "s", "e", "w", "ne", "nw", "se", "sw" или "center")	Какой угол или сторона размещаемого виджета будет указана в аргументах x/y/relx/rely. По умолчанию "nw" - левый верхний
	bordermode ("inside", "outside", "ignore")	Определяет, в какой степени будут учитываться границы при размещении виджета (пограничный режим (внутри, снаружи, игнорировать)).
	x и y	Абсолютные координаты (в пикселях) размещения виджета
	width и height	Абсолютные ширина и высота виджета
	relx и rely	Относительные координаты (от 0.0 до 1.0) размещения виджета
	relwidth и relheight	Относительные ширина и высота виджета
	sticky ("n", "s", "e", "w" или их комбинация)	Указывает: к какой границе "приклеивать" виджет. Позволяет расширять виджет в указанном направлении. Границы названы в соответствии со сторонами света. "n" (север) - верхняя граница, "s" (юг) - нижняя, "w"


Виджет	Аргументы	Значения
		(запад) - левая, "е" (восток) - правая

### Индивидуальное задание

№ варианта	Задания	Варианты решения (Придумайте свои!)
0.	<p>Напишите программу, которая переводит введенное десятичное число в двоичное, восьмеричное или шестнадцатеричное.</p> <p>Выбор системы счисления оформите через группу радиокнопок.</p> <p>Ввод числа оформите через однострочное текстовое поле, вывод — либо как метка, либо однострочное текстовое поле</p>	
1.	<p>Написать простейший калькулятор, состоящий из текстового поля для ввода чисел, и кнопок "+", "-", "*", "/", ".".</p> <p>Результат вычислений должен отобразиться в метке.</p> <p>Если арифметическое действие выполнить невозможно, то в метке должно появляться сообщение "ERROR"</p>	
2.	<p>Напишите программу, которая генерирует окно с тремя метками и двумя тестовыми полями и кнопкой.</p> <p>В метках содержится текст "Введите логин" и "Введите пароль".</p> <p>Кнопка с надписью "Проверка" инициирует проверку пароля.</p> <p>В третьей строке выводится текст "Доступ разрешен" или "Доступ запрещен"</p>	
3.	<p>Написать программу, которая переводит введенную сумму в рублях в три любые валюты.</p> <p>Выбор валюты выполняется радиокнопками.</p> <p>Ввод числа — однострочное текстовое поле.</p> <p>Вывод — метка.</p>	

№ варианта	Задания	Варианты решения (Придумайте свои!)
4.	<p>Напишите программу, состоящую из семи кнопок, цвета которых соответствуют цветам радуги. При нажатии на ту или иную кнопку в текстовое поле должен вставляться код цвета, а в метку – название цвета.</p> <p>Форматы цветов в практической работе 8 - Графика в Python</p>	
5.	<p>Напишите программу, которая переводит введенное число (в метрах) в мили, ярд, фут или дюйм.</p> <p>Выбор меры оформите через группу радиокнопок.</p> <p>Ввод числа оформите через однострочное текстовое поле, вывод — либо как метка, либо однострочное текстовое поле</p>	См. 3 вариант
6.	<p>Напишите программу, в которой имеется несколько объединенных в группу радиокнопок, индикатор которых выключен (<code>indicatoron=0</code>).</p> <p>Если какая-нибудь кнопка включается, то в метке должна отображаться соответствующая ей информация.</p> <p>Обычных кнопок в окне быть не должно.</p>	
7.	<p>Напишите программу, состоящую из двух списков Listbox.</p> <p>В первом будет список студентов, заданный программой.</p> <p>Второй для студентов - должников, изначально пуст.</p> <p>При клике на одну кнопку студент должен переходить из одного списка в другой.</p> <p>При клике на вторую кнопку – возвращаться.</p> <p>Предусмотрите возможность множественного выбора элементов списка и их перемещения.</p>	

№ варианта	Задания	Варианты решения (Придумайте свои!)
8.	<p>Напишите программу, которая генерирует окно с меткой и текстовым полем.</p> <p>После ввода пользователем текста в поле и нажатия Enter, введенный текст должен отображаться в метке.</p>	
9.	<p>Напишите программу по следующему описанию.</p> <p>Нажатие Enter в однострочном текстовом поле приводит к перемещению текста из него в список (экземпляр Listbox).</p> <p>При двойном клике (&lt;Double-Button-1&gt;) по элементу списка, он должен копироваться в текстовое поле.</p>	
10.	<p>Напишите программу, состоящую из семи кнопок, цвета которых соответствуют цветам радуги.</p> <p>При нажатии на ту или иную кнопку в текстовое поле должен вставляться код цвета, а в метку – название цвета.</p> <p>Коды цветов в практической работе 8 - Графика в Python</p>	
11.	<p>Напишите программу, состоящую из восьми кнопок, каждая из которых соответствует определенному шрифту.</p> <p>При нажатии на ту или иную кнопку в текстовое поле должна вставляться фамилия, выведенная соответствующим шрифтом</p>	См. вариант 10
12.	<p>Напишите программу, состоящую из двух списков Listbox.</p> <p>В первом будет список товаров, заданный программой. Второй для покупок, изначально пуст.</p> <p>При клике на одну кнопку товар должен переходить из одного списка в другой.</p> <p>При клике на вторую кнопку – возвращаться.</p> <p>Предусмотрите возможность множественного выбора</p>	

№ варианта	Задания	Варианты решения (Придумайте свои!)
13.	Напишите программу, в которой имеется объединенные в группу радиокнопки: Праздничные даты. Индикатор радиокнопок выключен (indicatoron=0). При включении кнопки в метке отображается наименование праздника	См. вариант 6
14.	Написать программу, в которой однострочное поле предназначено для ввода значения температуры по Фаренгейту. Виджет кнопки читает значение текстового поля и конвертирует его из значения по Фаренгейту в значение по Цельсию и после нажатия помещает результат в виде текста в ярлыке	

## 17. ЛИТЕРАТУРА

1. Шелудько, В. М. Основы программирования на языке высокого уровня Python : учебное пособие : [16+] / В. М. Шелудько. – Ростов-на-Дону ; Таганрог : Южный федеральный университет, 2017. – 147 с. : ил. – Режим доступа: по подписке. – URL: <https://biblioclub.ru/index.php?page=book&id=500056> (дата обращения: 21.03.2022). – Библиогр. в кн. – ISBN 978-5-9275-2649-9. – Текст : электронный.
2. Шелудько, В. М. Язык программирования высокого уровня Python: функции, структуры данных, дополнительные модули : учебное пособие : [16+] / В. М. Шелудько. – Ростов-на-Дону ; Таганрог : Южный федеральный университет, 2017. – 108 с. : ил. – Режим доступа: по подписке. – URL: <https://biblioclub.ru/index.php?page=book&id=500060> (дата обращения: 21.03.2022). – Библиогр. в кн. – ISBN 978-5-9275-2648-2. – Текст : электронный.
3. Буйначев, С. К. Основы программирования на языке Python : учебное пособие / С. К. Буйначев, Н. Ю. Боклаг ; Уральский федеральный университет им. первого Президента России Б. Н. Ельцина. – Екатеринбург : Издательство Уральского университета, 2014. – 92 с. : табл., ил. – Режим доступа: по подписке. – URL: <https://biblioclub.ru/index.php?page=book&id=275962> (дата обращения: 21.03.2022). – Библиогр. в кн. – ISBN 978-5-7996-1198-9. – Текст : электронный



## **18. ЗАКЛЮЧЕНИЕ**

В данном учебно-методическом пособии представлены задания на лабораторные работы по дисциплине «Программирование». Для каждой работы приведен основной теоретический материал, примеры, аналогичные заданиям. Студенты выполняют задания индивидуально по вариантам. После написания и отладки программы оформляется отчет, и работа защищается в установленном порядке.

Локальный электронный методический материал

Елена Юрьевна Заболотнова  
Светлана Александровна Калинина

ПРОГРАММИРОВАНИЕ

Редактор Г. А. Смирнова

Уч.-изд. л. 5,6. Печ. л. 5,6

Издательство федерального государственного бюджетного образовательного  
учреждения высшего образования  
«Калининградский государственный технический университет».  
236022, Калининград, Советский проспект, 1