

Документ подписан простой электронной подписью  
Информация о владельце:  
ФИО: Локтионова Оксана Геннадьевна  
Должность: проректор по учебной работе  
Дата подписания: 19.03.2021 12:43:08  
Уникальный программный ключ:  
0b817ca911e6668abb13a5d426d39e5f1c11eabbf73e943df4a4851fda56d089

# МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное  
образовательное учреждение высшего образования  
«Юго-Западный государственный университет»  
(ЮЗГУ)

Кафедра информационной безопасности

УТВЕРЖДАЮ:

Проректор по учебной работе

  
О.Г. Локтионова  
(подпись, инициалы, фамилия)

«15» 03



## ПРОГРАММИРОВАНИЕ НА ЯЗЫКЕ PYTHON

Методические указания по выполнению лабораторных работ по  
дисциплине «Методы программирования» для студентов всех форм  
обучения направления подготовки магистров  
10.05.02 «Информационная безопасность телекоммуникационных  
систем»

Курск – 2021

УДК 004.43

Составитель А.А. Чаплыгин

Рецензент

Кандидат технических наук, доцент Ефремова И.Н.

**Программирование на языке Python:** методические указания по выполнению лабораторных работ / Юго-Зап. гос. ун-т; сост.: А.А. Чаплыгин. Курск, 2021. 15 с.: Библиогр.: с.15.

Содержат сведения о языке Python, а также приведены примеры и задания для лабораторных работ.

Методические указания соответствуют требованиям программы по направлению подготовки бакалавров: 10.05.02 «Информационная безопасность телекоммуникационных систем».

Предназначены для студентов всех форм обучения направления подготовки бакалавров 10.05.02 «Информационная безопасность телекоммуникационных систем».

Текст печатается в авторской редакции

Подписано в печать Формат 60x84 1/16.

Усл. печ. л. 0,87 Уч. – изд. л. 0,78 .Тираж экз. Заказ 477. Бесплатно.

Юго - Западный государственный университет.

305040, г. Курск, ул. 50 лет Октября, 94.

## Введение

Язык Python был создан в 1991 году голландским программистом Гвидо ван Россумом. Это язык высокого уровня с динамической типизацией, поддерживающий объектно-ориентированное программирование. В настоящее время язык Python является одним из самых популярных языков, который используется не только для системных сценариев, но и для крупных проектов в различных областях, включая Web-разработку[1,2,3]. Простота языка позволяет использовать этот язык для обучения программированию[4].

## Арифметические операции.

Язык Питон включает в себя все арифметические операции над целыми и вещественными числами.

Сложение:

```
>>> 10 + 20
```

```
30
```

Вычитание:

```
>>> 30 - 40
```

```
-10
```

Умножение:

```
>>> 254 * 453
```

```
115062
```

Деление:

```
>>> 34.0 / 56
```

```
0.6071428571428571
```

Целочисленное деление:

```
>>> 68 // 3
```

```
22
```

Остаток от деления:

```
>>> 68 % 3
```

2

Возведение в степень:

```
>>> 32 ** 22
```

```
1298074214633706907132624082305024L
```

В языке Питон можно оперировать с любыми даже самыми большими числами, ограничений нет.

### **Переменные. Ввод и вывод.**

В языке Питон переменные не нужно объявлять, им сразу присваивают значение, при этом тип переменной определяется динамически в зависимости от типа выражения.

```
a = 10
```

Создается переменная a и ей присваивается целое значение 10

```
y = a ** 2 + 4 * a + 5
```

Создается переменная y и ей присваивается вычисленное значение выражения со значением a, которое было присвоено раньше.

Переменные могут иметь русское название:

```
число = 25
```

```
квадрат = число ** 2
```

Строковые значения заключаются в одинарные или двойные кавычки, лучше использовать одинарные.

```
строка = 'Привет'
```

Здесь создается переменная со строковым типом.

Вещественная переменная:

```
e = 2.7
```

Преобразование типов осуществляется через встроенные функции:

```
a = '10'
```

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

Функция int преобразует тип из строки в целый.

Функция float преобразует тип из строки в вещественный.

```
>>> float('4.5')
```

```
4.5
```

Функция `str` преобразует тип из числа в строку.

```
>>> str(1)
```

```
'1'
```

```
>>> str(1.5)
```

```
'1.5'
```

Переменные логического типа принимают значения истина или ложь:

```
flag = True
```

```
notFlag = False
```

Функция `print()` выводит значение выражения. Это может быть переменная или константа. Можно вывести несколько значений, они указываются через запятую.

```
print(a)
```

```
print(a, b, c)
```

При этом автоматически вставляются пробелы между значениями и перевод строки в конце.

```
a = 10
```

```
b = 20
```

```
print('a =', a, 'b =', b)
```

выведет:

```
a = 10 b = 20
```

Специальные параметры могут изменить значения по умолчанию.

```
print(a, end=' ')
```

вместо перевода строки в конце выведется пробел (то есть заданный символ или строка)

```
print(10, 20, sep=',')
```

вместо пробела между значениями выведется запятая:

10,20

Ввод значений из консоли осуществляется функцией `input`:

```
a = input()
```

Здесь переменная `a` принимает строковое значение, которое вводит пользователь без перевода строки.

### **Циклы и условия.**

В языке Питон существуют следующие циклы: итерационный цикл `for` и цикл с предусловием `while`.

Цикл `for` используется если необходимо повторить какое-то действие заданное число раз.

```
for i in range(10):
```

данный цикл повторяет тело цикла 10 раз со значением итерационной переменной `i` от 0 до 9

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

`i` принимает значения 1, 2, 3, 4

```
for i in range(0, 10, 2):
```

`i` принимает значения: 0, 2, 4, 6, 8

выражения в `range` для начального значения, верхней границы и шага могут быть любыми

Операторы языка которые должны выполняться внутри цикла следуют за оператором `for` и отделяются одним отступом(табуляцией):

```
for i in range(10):
```

```
    print(i)
```

Все операторы внутри цикла должны быть с одинаковым отступом:

```
for i in range(10):
```

```
    a = input()
```

```
    print(i, a)
```

Условие в языке Питон выглядит следующим образом:

```

if <условие1>:
    <блок операторов по истине условия1>
else if <условие2>::
    <блок операторов по истине условия2>
else:
    <блок операторов иначе>

```

Как и в циклах операторы внутри условий должны быть отделены отступами.

```

if a == 0:
    b = 10

```

Виды сравнений:

равно ==

не равно !=

больше >

меньше <

больше или равно >=

меньше или равно <=

Составные условия соединяются при помощи and(и), or(или), not(не):

```

>>> a = 1
>>> b = 1
>>> if a == 1 and b == 1:
...     print(a, b)
1 1
>>> if not a == 0:
...     print(a)
1
>>> if a == 1 or b == 0:
...     print(a)

```

...

1

Цикл `while` используется когда необходимо повторять некоторые действия пока условие истинно:

```
while n > 0:
    n = n — 1
```

Для принудительного выхода из цикла используется оператор `break`:

```
while n > 0:
    n = n — 1
    if n == 3:
        break
```

Важно в точности соблюдать отступы, внутренние блоки отделяются отступами от внешних.

### **Работа со списками.**

Списки в Питоне представляют собой основную структуру данных. Список содержит последовательность элементов разных типов. Он заменяет собой массив.

список из трех чисел:

```
a = [1, 2, 3]
```

список строк:

```
b = ['aa', 'bb', 'cc']
```

К элементам списка можно обратиться по индексу:

```
>>> a[1]
```

```
2
```

```
>>> b[2]
```

```
'cc'
```

Индекс списка начинается от 0. Длина списка определяется функцией `len`:

```
>>> len(a)
```

3

Добавление элементов в список происходит с помощью метода `append`:

```
>>> a.append(10)
```

```
>>> a
```

```
[1, 2, 3, 10]
```

Срез списка — это часть элементов списка:

```
>>> a[1:3]
```

```
[2, 3]
```

В срезе указывается начальный индекс и конечный индекс, при этом начальный индекс попадает в срез, а конечный — нет.

Если начальный индекс не указан — срез берется от начала списка (в этом случае второй индекс означает количество элементов):

```
>>> a[:2]
```

```
[1, 2]
```

Если конечный индекс не указан, то срез берется до конца списка:

```
>>> a[1:]
```

```
[2, 3, 10]
```

Отрицательный индекс означает, что элемент берется с конца: последний элемент:

```
>>> a[-1]
```

```
10
```

Срез, исключая последний элемент

```
>>> a[1:-1]
```

```
[2, 3]
```

В цикле `for` можно обойти все элементы списка:

```
for i in a:
```

```
    print(i)
```

При этом итерационная переменная принимает последовательно значения всех элементов списка.

Специальное условие `in` служит для проверки нахождения элемента в списке:

```
if <элемент> in <список>:
```

Например:

```
if 1 in a:
```

```
    print(1)
```

### **Работа со строками.**

Строка в Питоне представляет собой список символов:

```
>>> a = 'str'
```

```
>>> for c in a:
```

```
...     print(c)
```

```
s
```

```
t
```

```
r
```

К строке можно обращаться по индексу

```
>>> a[2]
```

```
r
```

Также можно брать срез:

```
>>> a[1:]
```

```
'tr'
```

### **Работа с файлами.**

Для того чтобы создать или открыть файл нужно использовать функцию `open`. Она возвращает объект — дескриптор файла, с помощью которого ведется дальнейшая работа с файлом.

```
f = open(имя_файла, режим)
```

Режимы открытия файла:

'r' — для чтения текстового файла,  
'rb' — для чтения двоичного файла,  
'w' — создание текстового файла,  
'wb' — создание двоичного файла,  
'a' — открытие текстового файла для добавления строк в конец файла,  
'ab' - открытие двоичного файла для добавления байт в конец файла,  
'r+t' — для чтения и записи текстового файла,  
'r+b' — для чтения и записи двоичного файла.

Например:

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

При работе с текстовыми файлами в файл пишутся или из файла читаются строки и символы. В двоичном режиме работа идет со списками байт.

Чтение файла целиком:

```
data = f.read()
```

Чтение заданного количества символов или байт:

```
data = f.read(count), где count — число символов или байт.
```

Запись в файл:

```
f.write(data), где data — строка и список байт для записи.
```

Перемещение указателя файла:

```
f.seek(pos), где pos — смещение в файле относительно начала
```

Закрытие файла (необходимо в конце работы с файлом):

```
f.close()
```

Процедуры и функции.

Процедуры и функции в Питоне объявляются следующим образом:

```
def имя(параметры):
```

блок операторов

Имена параметров перечисляются через запятую:

```
def func(param1, param2):
```

Функция без параметров:

```
def func()
```

Тело функции — блок операторов отделяется отступом:

```
def func():
```

```
    a = 2
```

```
    b = 3
```

Для выхода из функции используется оператор `return`, который возвращает значение из функции (для процедур — без значения):

```
def search(l, elem):
```

```
    for e in l:
```

```
        if e == elem:
```

```
            return True
```

```
    return False
```

### **Задания для лабораторной работы.**

1. Напишите программу, которая выводит числа от 1 до 10.
2. Напишите программу, которая выводит значение функции  $y = x^2 + 4x + 10$  для  $x$ , введенного пользователем
3. Напишите программу, которая выводит таблицу значений функции  $y = 2x^2 + 3x - 5$ , для значений  $x$  от 0 до 20
4. Напишите программу, которая создает список чисел, добавляет туда несколько чисел и выводит этот список, используя цикл.
5. Напишите программу, которая в цикле ожидает ввод от пользователя и выводит введенную строку. Цикл прекращается, когда будет введена пустая строка.
6. Напишите программу, которая заполняет список чисел введенными пользователем числами и затем печатает этот список.

7. Напишите программу, которая создает список строк и выводит только четные элементы этого списка.

8. Напишите программу, которая ожидает ввода строки и затем выводит последние 5 символов этой строки.

9. Напишите программу, которая реализует поиск строки с помощью функции в введенном пользователем списке.

10. Напишите программу, которая создает текстовый файл и записывает туда введенные пользователем строки.

11. Напишите программу, которая читает содержимое тестового файла и выводит его в консоль.

12. Напишите программу, которая читает двоичный файл и выводит побайтно содержимое этого файла.

13. Напишите программу, которая подсчитывает число строк и символов в текстовом файле.

14. Напишите программу, которая соединяет два текстовых файла в один, и записывает третий файл.

15. Напишите программу, которая подсчитывает количество слов разделенных пробелами в текстовом файле.

16. Напишите программу, которая осуществляет поиск слова в текстовом файле и выводит строку файла, которая содержит заданное слово.

17. Напишите программу, которая меняет переводы строк в текстовом файле на пробелы и выводит измененные строки в консоль.

18. Напишите программу для простого калькулятора, который может складывать, вычитать, умножать и делить числа.

19. Напишите программу, которая осуществляет проверку введенного имени пользователя и пароля. Правильные имена и пароли задаются в текстовом файле.

20. Напишите программу, которая шифрует текстовый файл шифром Цезаря, сдвигая каждый код символа на 2.

### **Контрольные вопросы.**

1. Какие типы программирования поддерживает язык Питон?

2. Какие арифметические операции есть в языке Питон?
3. Как объявить переменную в Питоне?
4. Какие основные типы данных есть в языке Питон?
5. Каким образом вывести значение выражения в консоль на языке Питон?
6. Какая функция служит для ввода значений на Питоне?
7. Какая функция служит для конвертации строки в целое число?
8. Какая функция служит для конвертации строки в вещественное число?
9. Какая функция служит для конвертации числа в строку?
10. Когда применяется цикл for?
11. Когда применяется цикл while?
12. Как выглядит условный оператор на языке Питон?
13. Какие виды условий есть в языке Питон?
14. Как комбинируются условия в языке Питон?
15. Что такое список в языке Питон?
16. Когда применяются списки в программе?
17. Какой метод добавляет элемент в список?
18. Что такое срез списка?
19. Каким образом на языке Питон можно получить часть строки?
20. Какие функции работы с файлами есть в языке Питон?
21. Какие виды файлов существуют?
22. Какие режимы открытия файлов есть в языке Питон?
23. Какой тип данных возвращается в результате чтения текстового файла?
24. Какой тип данных возвращается в результате чтения двоичного файла?
25. Каким образом объявляется функция на языке Питон?
26. Какой оператор возвращает значение функции?

## Список литературы.

1. Лутц М. Изучаем Python, 4-е издание. – Пер. с англ. – СПб.: Символ-Плюс, 2011. – 1280 с.
2. Златопольский Д.М. Основы программирования на языке Python. – М.: ДМК Пресс, 2017. – 284 с.
3. Федоров, Д. Ю. Программирование на языке высокого уровня Python : учебное пособие для прикладного бакалавриата / Д. Ю. Федоров. – 2-е изд., перераб. и доп. – Москва : Издательство Юрайт, 2019. – 161 с. – (Бакалавр. Прикладной курс). – ISBN 978-5-534-10971-9. – Текст: электронный // ЭБС Юрайт [сайт]. – URL: <https://urait.ru/bcode/437489> .
4. Доусон М. Програмируем на Python. – СПб.: Питер, 2014. – 416 с.