Национальный исследовательский университет «МИЭТ»

Институт интегральной электроники (группа ЭН-14М, каф. ПКИМС)

Компьютерные технологии в научных исследованиях

Лабораторная работа №4

Основы языка Python

Создание скрипта

Создание файла

```
touch ./script.py
```

Написание кода

```
#!/usr/bin/python
print("Hello from python! :)")
```

Установка прав

```
chmod +x ./script.py
```

Запуск

```
./script.py
```

Форматированный вывод: новый синтаксис (Python3)

Использование форматирования в функции print:

```
s1 = 'A'
s2 = 'B'

print("Values: {} {}".format(s1, s2))


print("Values: {0} {1}".format(s1, s2))


print("Values: {1} {0}".format(s1, s2))


print("Values: {a} {b}".format(a=s1, b=s2))
```

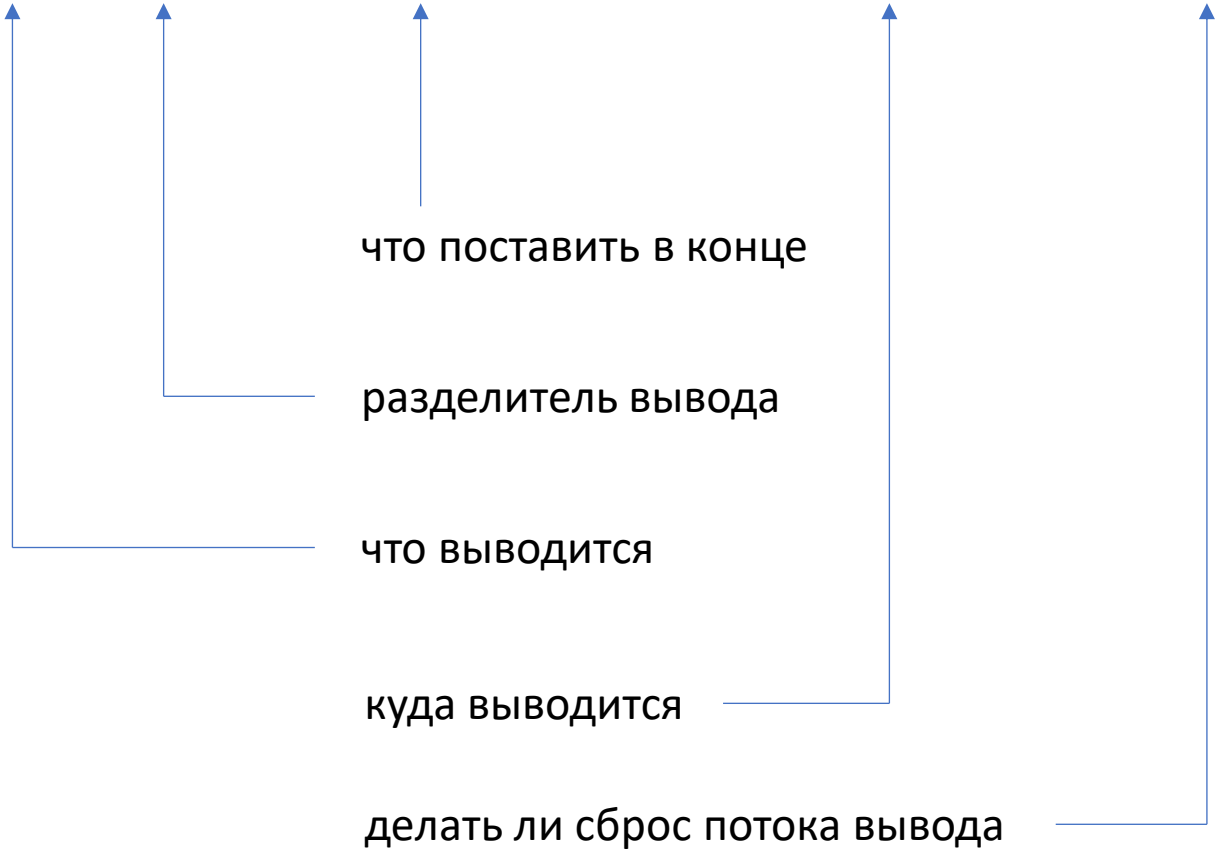
Получение информации о типе данных

```
x=1
y=3.1415

print("Тип X='{}'\nТип Y='{}'".format( type(x), type(y) ))
```

Аргументы функции print

```
print(*items, sep=' ', end='\n', file=sys.stdout, flush=False)
```



Чтение данных с консоли (Python3)

```
val = input("Enter value:")

print("Val={}\nVal type={}".format(val, type(val)))
```

Функции приведения типов:

```
str()
int()
long()
float()
```

Типы данных в Python

- **числа**
  - целые `x = 4`
  - вещественные `y = 3.1415`
  - комплексные `z = complex(1, 4)`
- **строки** `'this is a string'`
- списки `list_var = ['this', 'is', 'a', 'string']`
- множества `set_var = {'this', 'is', 'a', 'string'}`
- словари `dict_var = {'a' : 'this', 'b' : 'is', 'c' : 'a', 'd' : 'string'}`
- кортежи `tuple_var = ('this', 'is', 'a', 'string')`

Операции над числами (1)

Математические операции над числами

Операция	Значение
`x + y`	Сложение
`x - y`	Вычитание
`x * y`	Умножение
`x / y`	Деление
`x // y`	Получение целой части от деления
`x % y`	Остаток от деления
`-x`	Смена знака числа
`abs(x)`	Модуль числа
`divmod(x, y)`	Пара (x // y, x % y)
`x ** y`	Возведение в степень
`pow(x, y[, z])`	x^y по модулю (если модуль задан)

Операции над числами (2)

Побитовые операции над **целыми** числами:

Операция	Значение
`x | y`	Побитовое *или*
`x ^ y`	Побитовое *исключающее или*
`x & y`	Побитовое *и*
`x << n`	Битовый сдвиг влево
`x >> y`	Битовый сдвиг вправо
`~x`	Инверсия битов

Операции над числами (3)

```
x = 4131

print("Dec x = {}, Bin x = {}".format( x, bin(x) ))
x = x >> 2
print("Dec x = {}, Bin x = {}".format( x, bin(x) ))

y = 2.71828
res1 = x * y
print(res1)

c = complex(2, -3)
res2 = c*(x - y)
print(res2)
```

```
E:\>python test.py
Dec x = 4131, Bin x = 0b1000000100011
Dec x = 1032, Bin x = 0b10000001000
2805.26496
(2058.56344-3087.84516j)
```

Работа со строками

```
strval = "PKIMS rulez!"
n = len(strval)
print (n)

print(strval[2])

print(strval[1:])

print(strval[1:4])

print(strval[-3])

print(strval[-3:])

print(strval[3:-3])
```

Некоторые строковые методы

Метод	Описание
<str>.center(<width>)	возвращает копию <str>, выровненную по центру до <width>
<str>.count(<sub> [, <start> [, <end>]])	Считает число вхождений подстрок <sub> с возможным указанием диапазона
<str>.startswith(<sub>) <str>.endswith(<sub>)	Возвращает True, если строка начинается на <sub> / оканчивается на <sub>
<str>.find(<sub> [, <start> [, <end>]])	Ищет <sub> в строке <str> с возможным указанием диапазона, возвращает индекс или -1
<str>.isalpha(), <str>.isdigit()	Проверяет строку на то, что она является только символьной / только цифровой
<str>.upper(), <str>.lower()	Возвращает копию <str>, переведённую в верхний / нижний регистр
<str>.split([sep])	Разбивает строку по [sep], если не задан – по пробельным символам
<str>.replace(<old>, <new> [, <count>])	Замена подстрок в строке

Сырые строки в Python3

```
f = open("/home/topgun/test.txt", 'r')
text = f.read()
print(text)
```

```
f = open("D:\\Test\\test.txt", 'r')
text = f.read()
print(text)
```

```
f = open("D:/Test/test.txt", 'r')
text = f.read()
print(text)
```

```
f = open(r"D:\Test\test.txt", 'r')

text = f.read()

print(text)
```

Условный оператор if

```
if var1 == var2:
    print ("var1 = var2")
else:
    print ("var1 != var2")
```

```
if var1 > var2 and var2 < var3:
    print ("var2 is in the middle")
```

```
if var1 == var2:
    print ("var1 = var2")
elif var1 < var2:
    print ("var1 < var2")
else:
    print ("var1 > var2")
```

Операторы сравнения:

==
!=
<
>
<=
>=

Логические операторы:

and
or
not

Списки в Python

```
l = [1, 2, 3, 4, 5, 6]
print(l)

for i in l:
    print (i)
```

```
l = []
print(l)

l.append(4)
print(l)
```

```
s = "ПКИМС рулит"
l = list(s)
print(l)
```

```
l = [1, 2, 3, 4, 5]

print(len(l))
print(l[0])
```

СПИСКИ, кортежи, множества, словари (1)

Список (List) – самая часто используемая структура данных в Python, аналог массивов с динамической типизацией в компилируемых языках программирования. Элементы индексируются.

```
l = list()
l = [ 1, 2, 3, 4, 5 ]

l.append(6)
l.append(5)

l[2]

l.count(5)

l.pop(0)

l.remove(4)

...
```

Списки, КОРТЕЖИ, множества, словари (2)

Кортеж (Tuple) – аналог списка, элементы которого нельзя изменить
Элементы индексируются.

```
t = tuple()
t = (1, 2, 3, 4, 5, 3, 4, 5)

t.count(3)

t.index(2)
t.index(2, 0)
t.index(2, 0, 4)
```

Списки, кортежи, МНОЖЕСТВА, словари (2)

Множество (Set) – набор уникальных элементов, порядок их следования не регламентирован.

```
s = set()
s = {1, 2, 3, 4, 5, 3, 4, 5}

s.add(6)

s.remove(5)

s.difference({3, 4})

s.intersection({3, 4, 5, 6})

s.symmetric_defference({3, 4, 5, 6})

...
```

Списки, кортежи, множества, СЛОВАРИ (4)

Словарь (Dictionary) – неупорядоченный список элементов с доступом по ключу.
Также – ассоциативные массивы, также хэш-таблицы.

```
d = dict()

d = {'name' : 'Dmitry', 'surname' : 'Bulakh'}

d['name’]

d.get('name')

d['name'] = 'Dmitiy'


...
```

Цикл for

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

```
for i in range(1, 10):
    print(i)
```

```
for i in range(1, 10, 2):
    print(i)
```

```
for i in range(20):
    print("%4d%20s" % (i, " is our value"))
```

```
for i in range(20):
    print("{:4} {:>20}".format(i, "is our value"))
```

Работа с файлами: чтение текстовых файлов

```
f = open("myfile.txt", 'r')

text = f.read()

print (text)

f.close()
```

```
f = open("myfile.txt", 'r')

for line in f:
  print (line)

f.close()
```

```
f = open("data.txt", 'r')

for line in f:
  for token in line.rstrip().strip():
    print(token)

f.close()
```

Работа с файлами: запись текстовых файлов

```
f = open("data.txt", 'w')
for count in range(1, 100, 5):
    f.write("Number is : ")
    f.write(str(count) + "\n")
f.close()
```

```
f = open("data.txt", 'w')
for count in range(1, 100, 5):
        print("Number is : {}".format(count), file=f)
f.close()
```

Использование модулей (1)

```
import os

print(os.getlogin())
print(os.getcwd())
```

```
import os
from sys import platform

print(os.name)
print(platform)
```

```
import os, sys

print(os.name)
print(sys.platform)
```

```
import os
import sys
from sys import platform

print(sys.argv)
print(os.name)
print(platform)
```

Использование модулей (2)

```
import os
import sys
from sys import platform


print(sys.argv)
print(os.name)
print(platform)
```

```
import sys
from sys import *


print(argv)
print(os.name)
print(platform)
```

```
from sys import argv as args

print(args)
```

Регулярные выражения: модуль re (1)

```
import re

ret = re.match('a+', 'aaababbccc')

if ret:
  print('Yes')
else:
  print('No')
```

```
if ret:
  print('Yes')
  print(ret.group())
else:
  print('No')
```

Регулярные выражения: модуль re (2)

```
import re

ret = re.fullmatch('a+', 'aaababbccc')

if ret:
  print('Yes')
  print(ret.group())
else:
  print('No')
```

```
import re

ret = re.match('b+', 'aaababbccc')

if ret:
  print('Yes')
  print(ret.group())
else:
  print('No')
```

Регулярные выражения: модуль re (3)

```
import re

ret = re.search('b+', 'aaababbccc')

if ret:
  print('Yes')
  print(ret.group())
else:
  print('No')
```

```
if ret:
  print('Yes')
  print(ret.group())
  print(ret.start())
  print(ret.end())
else:
  print('No')
```

Регулярные выражения: модуль re (4)

```
import re

ret = re.findall('b+', 'aaababbccc')

if ret:
  print('Yes')
  print(ret)
else:
  print('No')
```

```
if ret:
  print('Yes')
  print(ret)
  print(len(ret))
  print(ret[0])
else:
  print('No')
```

Функции: синтаксис вызова

```
def func1(a, b = 45):
  c = a+b
  return c

sum = func1(4)
print (sum)
```

```
def func1(a, b, strval):
  c = a+b
  print (strval)
  return c

sum = func1(strval ="String", a=4, b=6)
print sum
```

Создание и использование модулей

Файл module_re.py:

```
import re

def re_search(a, b):
  ret = re.search(a, b)
  if ret:
    return True
  else:
    return False
```

Файл main.py:

```
#!/usr/bin/python

import module_re

ret = module_re.re_search('b+', 'aaababbccc')

if ret:
  print("Yes")
else:
  print("No")
```

ООП в Python

```
class ClassName:

  var = "Simple var"

  def __init__(self, value):
    self.var = value

  def print_value(self):
    print ("Value=", self.value)


cls = ClassName("String value")

cls.print_value()
```

Библиотеки Python: PyQt5

PyQt5

```
import sys
from PyQt5.QtWidgets import QApplication, QWidget

if __name__ == '__main__':

    app = QApplication(sys.argv)

    win = QWidget()
    win.resize(250, 150)
    win.move(300, 300)
    win.setWindowTitle('Simple')
    win.show()

    sys.exit(app.exec_())
```