Программирование на языке Python

Повторение

Вывод на экран

Текст:

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

Значения переменных из памяти:

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

Арифметические выражения:

```
print ( a + 2*b )
```

Все вместе:

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

Подключение русского языка:

```
# coding: utf-8
```

Вывод на экран

С пробелами:

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

Без пробелов:

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

Без перехода на новую строку:

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

Ввод данных с клавиатуры

Символьная строка:

```
print( 'Введите имя:' )
s = input()
```

или так:

```
s = input('Введите имя:')
```

Целое число:

```
print( 'Введите целое число:' )
n = int(input())
```

или так:

```
n = int(input('Введите целое число:'))
```

Ввод данных с клавиатуры

Вещественное число:

```
print( 'Введите число:' )
x = float (input())
```

или так:

```
x = float (input('Введите число:'))
```

Ввод данных с клавиатуры

Два целых числа (каждое в отдельной строке):

```
print( 'Введите два числа:' )
a = int (input())
b = int (input())
```

в одной строке:

```
print( 'Введите два числа:' )
a, b = map(int, input().split())
```

```
input()              # "21 35"
input().split()      # ["21", "35"]
a = int("21")
b = int("35")
```

символьные строки

Присваивание

```
a = 6
b = 4
a = 2*a + 3*b
b = a / 2 * b
```

Сокращённая запись операций:

```
a += 1
b += a
a *= 2 + 3*b
b /= 2 * a
```

Остаток от деления – %

```
a = 1234
d = a % 10; print( d )
a = a // 10
d = a % 10; print( d )
a = a // 10
d = a % 10; print( d )
a = a // 10
d = a % 10; print( d )
a = a // 10 :
```

4

3

2

1

Условный оператор

```
if a > b:
    # что делать, если a > b
else:
    # что делать, если a <= b
```

отступы!

```
a = 12
if a > 20:
   a = 15
print ( a )
```

```
a = 12
if a > 2:
   a = 15
else:
   a = 8
print ( a )
```

Цепочка условий

```
cost = 1500
if cost<1000:
   print( "Скидок нет." )
elif cost<2000:
   print( "Скидка 2%." )
elif cost<5000:
   print( "Скидка 5%." )
else:
   print( "Скидка 10%." )
```

первое сработавшее условие

? Что выведет?

```
Скидка 2%.
```

Сложные условия

Задача: набор сотрудников в возрасте **25-40 лет** (включительно).

сложное условие

```
if v >= 25 and v <= 40 :
   print("подходит")
else:
   print("не подходит")
```

and «И»: **одновременное** выполнение всех условий!

Сложные условия

Задача: набор сотрудников в возрасте **25-40 лет** (включительно).

сложное условие

```
if v < 25 or v > 40:
   print("не подходит")
else:
   print("подходит")
```

or «ИЛИ»: выполнение **хотя бы одного** из двух условий!

Цикл с условием

```
k = 0
while k < 10:
   print ( "Привет" )
   k += 1
```

При каком условии заканчивает работу?

`k >= 10`

```
k = 10
while k > 0:
   print ( "Привет" )
   k -= 1
```

При каком условии заканчивает работу?

`k <= 0`

Цикл по переменной

```
for i in range(N):
    ...
```

сделай **N** раз

`range(N) = [0, 1, 2, …, N-2, N-1 ]`

`[0,1,2,3]`

N раз

```
for i in range(4):
    print(i)
```

```
0
1
2
3
```

Что выведет?

`[0,1,2,3,4]`

```
s = 0
for i in range(5):
    s += i
print(s)
```

```
10
```

Цикл по переменной

```
s = 0
for i in range(2,5):
    s += i
print(s)
```

от

до (не включая!)

[2,3,5)

s=2+3+4=9

9

"Бесконечный" цикл

```
s = 0
while True:
   x = int(input())
   if x == 0: break
   if x % 10 == 5:
      s += x
print( s )
```

выйти из цикла

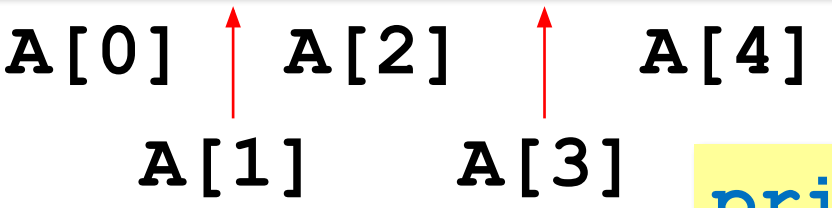
? Что плохо?

! Выход из цикла **while True** возможен только через оператор **break**!

Массивы (списки) в Python

Создание массива:

```
A = [1, 5, 0, -1, 12]
```




```
print(A[1])
```

5

```
print(2*A[0]+A[3])
```

1

```
A = 5*[0]
```

```
A = [0,0,0,0,0]
```

Вывод массива на экран

Как список:

```
print ( A )
```

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

В строчку через пробел:

```
for i in range(N):
  print ( A[i], end=" " )
```

```
1 2 3 4 5
```

пробел после вывода очередного числа

или так:

```
for x in A:
  print ( x, end=" " )
```

```
1 2 3 4 5
```

или так:

```
print ( *A )
```

↔

```
print (1, 2, 3, 4, 5)
```

разбить список на элементы

Заполнение случайными числами

```
from random import randint
A = []
for i in range(5):
  A.append(randint(1,6))
print(A)
```

наращиваем с каждым шагом

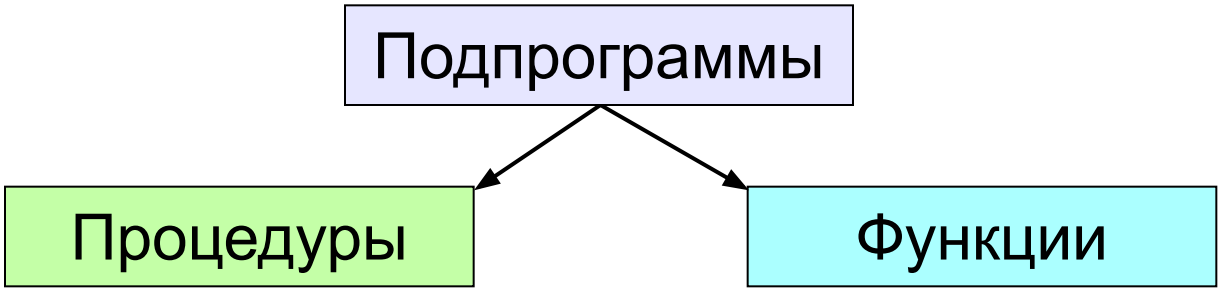
В чём отличие?

Или так:

```
from random import randint
A = 5*[0]
for i in range(5):
  A[i] = randint(1,6)
print(A)
```

сначала выделили память, потом меняем

Два типа подпрограмм



выполняют действия

+ возвращают некоторый результат

Процедура или функция?

а) рисует окружность на экране

б) определяет площадь круга

в) вычисляет значение синуса угла

г) изменяет режим работы программы

д) возводит число x в степень y

е) включает двигатель автомобиля

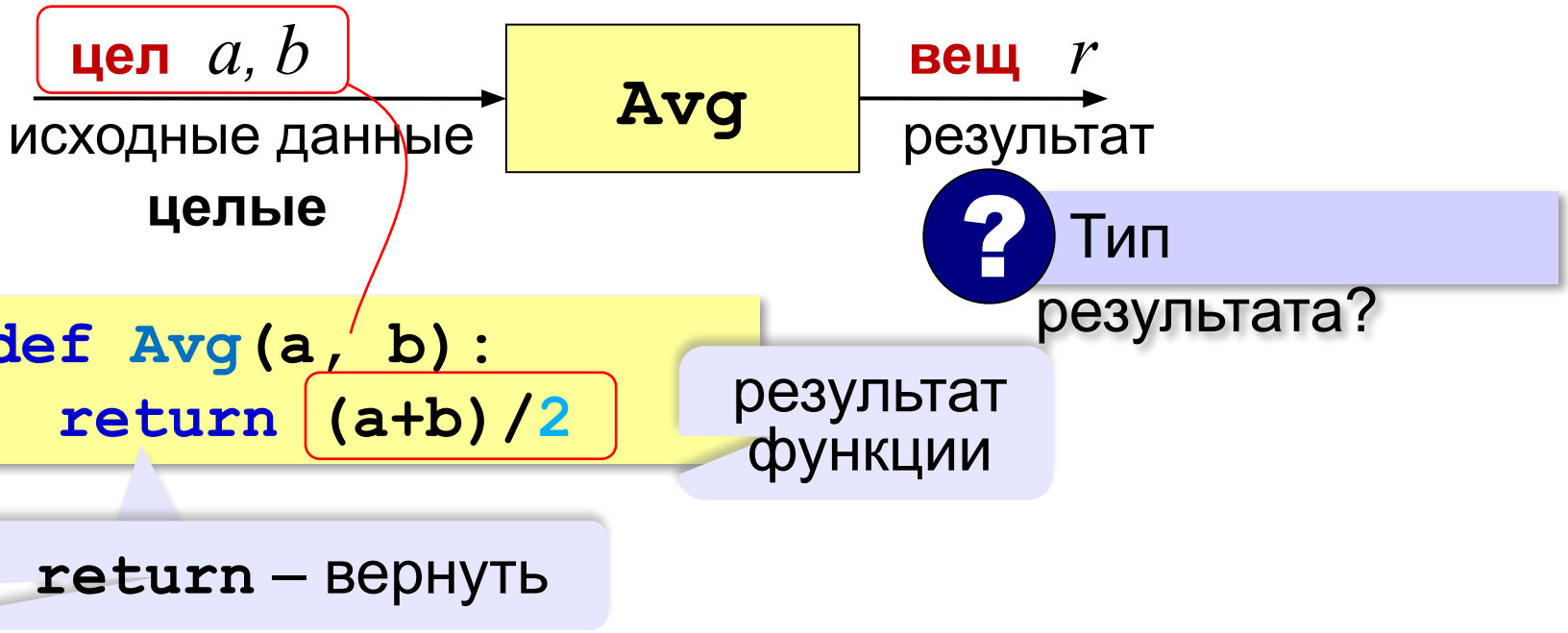
ж) проверяет оставшееся количество бензина в баке

з) измеряет высоту полёта самолёта

Что такое функция?

Функция — это вспомогательный алгоритм, который возвращает результат (число, строку символов и др.).

Задача. Написать функцию, которая вычисляет среднее арифметическое двух целых чисел.



Логические функции

Логическая функция — это функция, возвращающая логическое значения (**да** или **нет**).

- можно ли применять операцию?
- успешно ли выполнена операция?
- обладают ли данные каким-то свойством?

Логические функции

Задача. Составить функцию, которая возвращает «`True`», если она получила чётное число и «`False`», если нечётное.

```
def Even( N ):
  if N % 2 == 0:
    return True
  else:
    return False
```

```
def Even( N ):
  return (N % 2 == 0)
```

Сравнение строк

```
print("Введите пароль: ")
s = input()
if s == "sEzAm":
  print("Слушаюсь и повинуюсь!")
else:
  print("Пароль неправильный")
```

Какой правильный пароль?

Как одна строка может быть меньше другой?

стоит раньше в отсортированном списке

Сравнение строк

```
s1 = "паровоз"
s2 = "пароход"
if s1 < s2:
   print(s1, "<", s2)
elif s1 == s2:
   print(s1, "=", s2)
else:
   print(s1, ">", s2)
```

Что выведет?

```
паровоз < пароход
```

первые отличающиеся буквы

в < х!

Сравниваем с начала:

```
паровоз
пароход
```

«в»: код **1074**

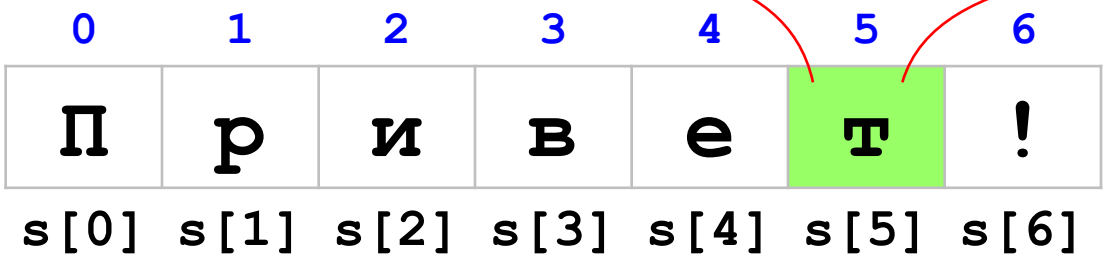
«х»: код **1093**

Обращение к символу по номеру

```
print ( s[5] )
```

```
print ( s[-2] )
```

s[len(s)-2]



! Символы нумеруются с нуля!

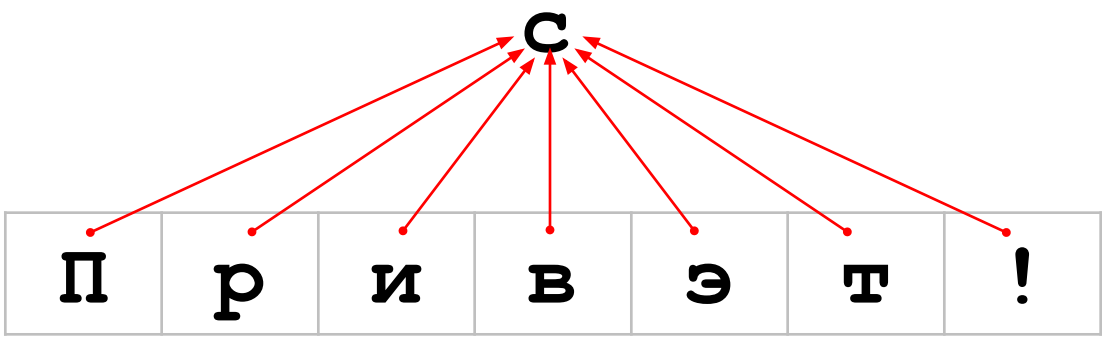
составить «кот»

```
s = "информатика"
kot = s[-2]+s[3]+s[-4]
```

Цикл перебора символов

```
sNew = ""
for c in s:
    if c == "э":
        sNew += "е"
    else:
        sNew += c
```

перебрать все символы строки



Операции со строками

Объединение *(конкатенация)* :

```
s1 = "Привет"
s2 = "Вася"
s  = s1 + ", " + s2 + "!"
```

"Привет, Вася!"

Умножение:

```
s = "АУ"
s5 = s*5
```

```
s5 = s + s + s + s + s
```

АУАУАУАУАУ

? Что получим?

Срезы строк (выделение части строки)

```
s = "0123456789"
s1 = s[3:8]       # "34567"
```

с какого символа

до какого (**не включая 8**)

```
s = "0123456789"
s1 = s[:8]        # "01234567"
```

от начала строки

```
s = "0123456789"
s1 = s[3:]        # "3456789"
```

до конца строки

Срезы строк

Срезы с отрицательными индексами:

```
s = "0123456789"
s1 = s[:-2]              # "01234567"
```

len(s)-2

```
s = "0123456789"
s1 = s[-6:-2]            # "4567"
```

len(s)-6 len(s)-2

Операции со строками

Удаление:

```
s = "0123456789"
s1 = s[:3] + s[9:]
```

"012" "9" "0129"

Вставка:

```
s = "0123456789"
s1 = s[:3] + "ABC" + s[3:]
```

"012" "3456789"

"012ABC3456789"

Поиск в строках

```
s = "Здесь был Вася."
n = s.find ( "с" )    # n = 3
if n >= 0:
   print ( "Номер символа", n )
else:
   print ( "Символ не найден." )
```

Находит первое слева вхождение подстроки!

Поиск с конца строки:

```
s = "Здесь был Вася."
n = s.rfind ( "с" )     # n = 12
```

Преобразования «строка» → «число»

Из строки в число:

```
s = "123"
N = int ( s )          # N = 123
s = "123.456"
X = float ( s )        # X = 123.456
```

Из числа в строку:

```
N = 123
s = str ( N )                 # s = "123"
s = "{:5d}".format(N)         # s = "  123"

X = 123.456
s = str ( X )                 # s = "123.456"
s = "{:7.2f}".format(X)       # s = " 123.46"
s = "{:10.2e}".format(X)      # s = "  1.23e+02"
```

Словари

Словарь (dictionary) в языке Python хранит коллекцию элементов, где каждый элемент имеет уникальный ключ и ассоциированное с ним некоторое значение.

Создание словаря

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


dictionary = {
    ключ1:значение1,
    ключ2:значение2,
    ...
}


objects = {}

objects = dict()
```

Преобразование списка в словарь

```
users_list = [
    ["Tom", "+111123455"],
    ["Bob", "+385563295"],
    ["Alice", "+956831256"],
]
user_dict = dict(users_list)
print(user_dict)  # {“Tom”: “+111123455”, “Bob”: “+385563295”, “Alice”: “+956831256”}
```

Получение и изменение словарей

```
users = {
    "Tom": "+111123455",
    "Bob": "+385563295",
    "Alice": "+956831256",
}

# получаем элемент с ключом
“Tom”
print(users["Tom"])    # +111123455

users["Bob"] = "+333333333"
print(users["Bob"])    # +333333333
```

Если при установки значения элемента с таким ключом не окажется, то произойдёт его добавление.

Удаление элементов

```
users = {
    "Tom": "+111123455",
    "Bob": "+385563295",
    "Alice": "+956831256"
}

del users["Alice"]
print(users)  # {"Tom": "+111123455", "Bob": "+385563295"}
```

Функции для работы со словарём

dict.pop(key)	Удаляет элемент по ключу key и возвращает удалённый элемент. Если элемента нет, то сгенерируется исключение KeyError
dict.clear()	Удаляет все элементы словаря
dict.copy()	Копирует содержимое словаря
dict.update(dict2)	Объединяет словарь dict2 с словарём dict
dict.items()	Возвращает набор кортежей. Каждый кортеж содержит ключ и значение элемента, которые при переборе мы тут же можем получить в переменные key и value.
dict.keys()	Возвращает ключи словаря
dict.values()	Возвращает значения словаря

Перебор словаря

```
users = {
    "Tom": "+111123455",
    "Bob": "+385563295",
    "Alice": "+956831256"
}
```

```
for key in users:
    print(f"User: {key}, Phone: {users[key]}")
```

```
for key, value in users.items():
    print(f"User: {key}, Phone: {value}")
```

```
for key in users.keys():
    print(key)
```

```
for value in users.values():
    print(value)
```

Комплексные словари

```
users = {
    "Tom": {
        "phone": "+971478745",
        "email": "tom12@gmail.com"
    },
    "Bob": {
        "phone": "+873690444",
        "email": "bob@gmail.com",
        "skype": "bob123"
    }
}
```

```
old_email = users["Tom"]["email"]
users["Tom"]["email"] = "supertom@gmail.com"
print(users["Tom"])
# { "phone": "+971478745", "email": "supertom@gmail.com }
```