

Занятие #2

Оператор For

Строки

Списки



Что напечатает?

```
print(123)
```

```
print('123')
```

```
print(123 == '123')
```

```
print(print(123))
```

```
print(123 + 123)
```

```
print('123' + '123')
```

```
print(type(123), type('123'))
```

```
print(123, '123')
```

```
print(print)
```



Задача 1-1

Ввести два числа x и y .

Напечатать сумму и произведение этих чисел (оператор $+$ и $*$)



Задача 1-2

Ввести два числа x и y .

Напечатать наибольшее из чисел $x + y$, $x - y$, $x * y$, x / y , $x // y$



Задача 1-3

Ввести два числа x и y .

Напечатать ВТОРОЕ ПО ВЕЛИЧИНЕ из чисел $x + y$, $x - y$, $x * y$, x / y , $x // y$



IF ... ELSE...

```
if x > 0:
```

```
    print('Положительное')
```

```
elif x == 0:
```

```
    print('Ноль')
```

```
else:
```

```
    print('Отрицательное')
```

```
#    > < >= <= == !=
```

```
#    and or not
```



Новые возможности оператора print

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

Попробуйте разные сочетания параметров:

sep – разделитель

end – окончание

Например:

```
print(1, 2, 3, sep = ":", end = ";")
```

```
print(1, end = "?")
```

```
print()
```

```
print(3)
```

Документация Python: <https://docs.python.org/3/> или

```
help(print)
```



Цикл for

for i in объект:

 группа операторов

Объект может быть строка, список и другие конструкции.

#Выполните операторы:

```
for i in 'Hello world':
```

```
    print(i)
```




Цикл for - примеры

```
for i in 'Hello world':
```

```
    print(i, end = " " )
```

```
for i in 'Hello world':
```

```
    print(i, end = "_" )
```

```
for i in 'Hello world':
```

```
    print(i, end = "" )
```



Функция range()

Функция range() применяется для генерации последовательности чисел.

`range(stop)` – чаще всего встречающийся вариант: 0, 1, 2, ... , stop - 1

`range(start, stop[, step])`

```
for j in range(5):
```

```
    print(j) # 0 1 2 3 4
```

```
for k in range(-5, 5):
```

```
    print(k, end = ', ') # -5, -4, -3, -2, -1, 0, 1, 2, 3, 4,
```

```
for n in range(5, -5, -2):
```

```
    print(n) # 5 3 1 -1 -3
```



Цикл `for i in range(10):`

Выполните оператор:

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

```
    print(i)
```

1. Доработайте эти строчки, чтобы для каждого `i` печатался также его квадрат
2. Доработайте эти строчки, чтобы для каждого четного `i` печатался `i – четное`, а для каждого нечетного `i` печатался `i – нечетное`
`0 – четное`
`1 – нечетное`
`2 – четное и т.д.`
(проверка на четность `if i % 2 == 0`)



Программа FizzBuzz, которую часто дают на собеседованиях

Напишите программу, которая запрашивает число n , а затем выводит на экран числа от 1 до n . Используйте `range(1, n + 1)`.

При этом вместо чисел, кратных трем, программа должна выводить слово «Fizz», а вместо чисел, кратных пяти — слово «Buzz».

Если число кратно и 3, и 5, то программа должна выводить слово «FizzBuzz»

Т.е. 1, 2, Fizz, 4, Buzz, Fizz, 7, и т.д.

(проверка на делимость на 3 `if i % 3 == 0`)



Цикл for + break, continue, else

for i **in** объект:

 группа операторов

 if проверка: break # выход из цикла

 if проверка: continue # переход в начало цикла

else:

 Операторы # ветка else выполняется если не было выхода с помощью оператора break

Объект может быть строка, список и другие конструкции.

Выполните оператор:

```
for I in 'Hello world':
```

```
    print(i)
```



Оператор break

```
>>> for i in 'hello world':
```

```
...     if i == 'o':
```

```
...         break
```

```
...     print(i * 2, end="")
```

```
...
```

```
#hheellll
```

```
#Выполните эту программу
```



Оператор continue

```
>>> for i in 'hello world':
```

```
...     if i == 'o':
```

```
...         continue
```

```
...     print(i * 2, end="")
```

```
...
```

```
#hheellll wwrrlldd
```

```
# выполните эту программу
```




Волшебное слово else

```
>>> for i in 'hello world':  
...     if i == 'a':  
...         break  
... else:  
...     print('Буквы а в строке нет')  
...
```

#Буквы а в строке нет

#Выполните эту программу



Программа, которая печатает только нечетные числа
а если k становится больше 10, то заканчивается цикл

```
n = int(input()) # Введите любое число больше 10  
for k in range(n):  
    if k > 10:  
        break  
    elif k % 2 == 0:  
        continue  
    else:  
        print(k)
```

#А что будет если убрать else и сдвинуть print налево?



Коллекции

- | | |
|--------------------|---------------------------------------|
| 1. Строка (str) | 'Hello world' |
| 2. Список (list) | [1, 100, 1, 'a', True] |
| 3. Кортеж (tuple) | (1, 100, 1, 'a', True) |
| 4. Словарь (dict) | {1:1, 22:100, 123:1, 'a':'a', 5:True} |
| 5. Множество (set) | {1, 100, 'a', True} |

Функция `type()`



Строки- последовательности символов Unicode

Строка задается либо парой одинарных ' ', либо двойных " " кавычек.

Существенной разницы в Python между одинарными и двойными кавычками нет.

```
>>> name = "" → пустая строка
```

```
>>> name = " → пустая строка
```

Не забывайте кавычки, без кавычек строка воспринимается как переменная!!!

```
print(hello)
```

Что напечатается?

Индексация строки

Для получения символа в строке нужно обратиться по индексу позиции.

Индексация строк начинается с 0

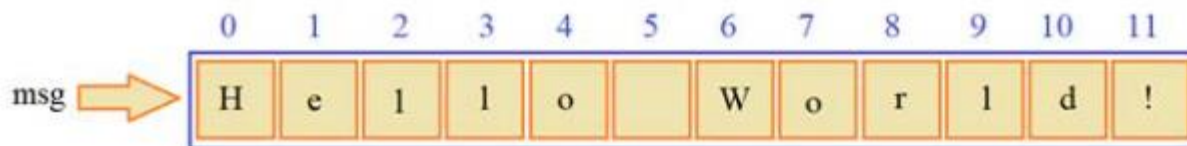
```
msg = "Hello World!"
```

```
print(msg[0]) → "H"
```

```
print(msg[0:1]) → "H"
```

```
print(msg[0:2]) → "He"
```

```
print(msg[0:5]) → "Hello"
```



Индексация строки

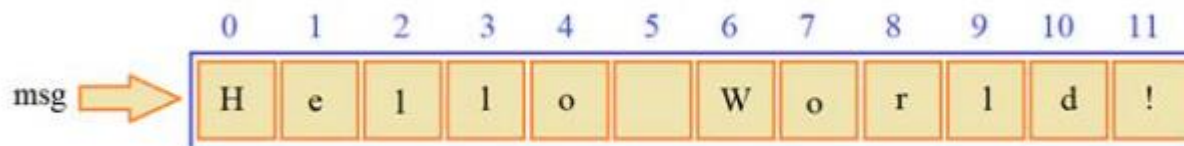
Дана строка `msg = "Hello World!"`

Введите три числа `p, q, r`

Напечатайте срез строки `msg[p:q:r]`.

Попробуйте ввести разные варианты: (0, 5, 1), (5, 0, -1), (1, 1, 1) и напечатать

любые другие варианты, отрицательные числа.





Напечатать треугольник из плюсов (2 варианта)

Ввести число n

Напечатать треугольник из символов + (пример для $n = 5$):

```
+  
++  
+++  
++++  
+++++
```

В цикле используйте оператор `print"+" * i`, где i переменная цикла

Попробуйте использовать два цикла, один вложенный в другой.

Первый для строк, а второй для печати "+" в строке



Список(List)

- Список — это упорядоченный набор элементов, перечисленных через запятую, заключённый в квадратные скобки
- Элементы списка могут быть разных типов, как правило, используются списки из элементов одного типа
- Список может содержать одинаковые элементы, в отличие от множества (set)
- Список можно изменить после создания, в отличие от кортежа (tuple)
- Список может содержать другие списки

Пример: [1, 11, 111, 0, -5, 'abc', [1,2], True, None]



Список (mutable - изменяемый)

Создать список можно двумя способами:

1. Вызывать функцию `list()`

```
lst = list()
```

2. Использовать квадратные скобки

```
lst = [] → Задали пустой список
```

```
lst = list([1, 4, 5])
```

```
lst = [1, 4, 5]
```

```
s = list("hello")
```

```
s = ["hello"]
```


Элементы списка разных типов. Индексы

Пример:

```
>>> lst = [10, True, [1,2], "abcdrfg"]
```

Индекс	0	1	2	3
Элемент списка	10	True	[1,2]	"abcdrfg"

```
>>> type(lst)
```

```
<class 'list'>
```



Список – изменяемый тип данных

Так как список - изменяемый тип данных, т.е. мы можем заменить любой элемент в нём на другой.

```
>>> mass = [4, 3, 2, 1]
```

```
>>> mass[0] = 8
```

```
>>> print(mass) #[8,3,2,1]
```



Получить размер списка: len()

```
>>> mass = [1,2,3]
```

```
>>> len(mass)
```

```
3
```

Чему равна длина пустого списка? []



Как нам напечатать элементы списка?

```
mass = [4, 5, 2, 3, 8, -1, 9]
```

Способ 1.

```
for j in mass:
```

```
    print(j)
```

Способ 2.

```
for j in range(len(mass)):
```

```
    print(mass[j])
```

Какой способ лучше?



Функция range() для генерации списка

Функция range() применяется для генерации последовательности чисел.

range(stop) – чаще всего встречающийся вариант: 0, 1, 2, ... , stop - 1

range(start, stop[, step])

```
print(list(range(5)))
```

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


```
print(list(range(-5, 5)))
```

```
[-5, -4, -3, -2, -1, 0, 1, 2, 3, 4]
```

```
print(list(range(5, -5, -2)))
```

```
[5, 3, 1, -1, -3]
```

```
print(type(range(5)))
```



Итерация списка с использованием For с проверкой типа элемента списка

```
input_list = [10, "э", 15, "ABC", 1]
```

```
for x in input_list:
```

```
    if type(x) == str:
```

```
        print(x * 2)
```

```
    elif type(x) == int:
```

```
        print(x ** 2)
```

Что будет напечатано?

Итерация списка с использованием for + enumerate

```
input_list = [10, "S", 15, "A", 1]
for k, v in enumerate(input_list):
    print(f"k={k} v={v}")
```

Вывод:

```
k=0  v=10
k=1  v=S
k=2  v=15
k=3  v=A
k=4  v=1
```



Метод append(x)

Добавление элемента в список осуществляется с помощью метода append()

```
>>> a = []
```

```
>>> a.append(3)
```

```
>>> a.append("hello")
```

```
>>> print(a)
```

```
[3, 'hello']
```




Поэлементный ввод списка из 5 элементов

1. Создайте пустой список `lst = []`
2. Далее цикл из 5 повторений (`for i in range(5):`)
3. В цикле ввод числа (`a = int(input())`)
4. Добавьте `a` в конец списка `lst` (`lst.append(a)`)
5. Печатайте список `lst`



Проверка нахождения значения в списке in , not in

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

```
>>> b = 2
```

```
>>> b in a
```

```
True
```


```
>>> 19 not in a
```

```
True
```

```
>>> 1 not in a
```

```
False
```

Очень медленная проверка, если список большой!!!



Вводите значения и проверяйте входят ли они в список

1. Дан список `lst = [11, 22, 33, -11, 0, 117]`
2. Создайте цикл из 5 итераций
3. На каждой итерации введите число `a`
4. Печатайте результат проверки, входит ли `a` в список `lst`

Используйте оператор `print(a in lst)`



Сложение списков

При сложении происходит объединение множеств массива

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

```
>>> b = [3,4]
```

```
>>> c = a + b
```

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



Присвоение списка

В случае, если вы выполните простое присвоение списков друг другу, то переменной **b** будет присвоена ссылка на тот же элемент данных в памяти, на который ссылается **a**

```
>>> a = [1, 3, 5, 7]
```

```
>>> b = a
```

```
>>> b[0] = 10
```

```
>>> print(a)
```

```
[10, 3, 5, 7]
```



Размножение списка

Мультипликация списка происходит при умножение его на число.

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

```
>>> b*2
```

```
[1,2,1,2]
```



Создание копии(клона) списка

```
>>> a = [1, 3, 5, 7]
```

```
>>> b = a[:]
```

Вопрос!

Что за оператор `[:]` ?

Второй способ:

```
>>> a = [1, 3, 5, 7]
```

```
>>> b = list(a)
```



Задание

Дан список чисел, например [1, 2, 3, 4, 5]

Создайте новый список, каждый элемент которого является суммой чисел первого списка с накоплением, т.е. [1, 3, 6, 10, 15]



Задание

Дан список списков чисел.

Сосчитайте сумму всех чисел всех подсписков.

Например:

`[[1,2,3], [10, 20], [100]] -> 136`



Задача 2-1

Введите число n . Напечатайте «таблицу умножения» на число n .

Например, для $n = 5$

$$1 \times 5 = 5$$

$$2 \times 5 = 10$$

$$3 \times 5 = 15$$

$$4 \times 5 = 20$$

$$5 \times 5 = 25$$

$$6 \times 5 = 30$$

...

$$9 \times 5 = 45$$



Задача 2-2

Введите список `lst`, состоящий из чисел.

Найдите и напечатайте наименьшее число из списка `lst`.

В Python есть функция `min`, которая решает эту задачу.

Но напишите свою программу, которая не использует функции `min`, `max`, `sort`, `sorted`.



Задача 2-3

Введите число n .

Сосчитайте и напечатайте факториал числа n !

$$n! = 1 * 2 * 3 * 4 * 5 * \dots * n$$

Не используйте, пожалуйста, функцию `math.factorial()`

(можно использовать ее для отладки)))