

# Занятие #3

Строки

Списки

Оператор while

Кортежи

# Что напечатает?

```
x = 123
```

```
for i in x:
```

```
    print(i)
```

a) 1 2 3

b) 123

**c) error**

**d) none of the mentioned**

```
#####
```

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

```
    if i == 5:
```

```
        break
```

```
    else:
```

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

```
else:
```

```
    print("Here")
```

a) 0 1 2 3 4 Here

b) 0 1 2 3 4 5 Here

c) 0 1 2 3 4

d) 1 2 3 4 5

# 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 `max`, `min`, `sort`, `sorted`.

А как найти второе по величине число?

А что если наименьших чисел несколько, как их всех напечатать?

## 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 )))

# 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])
```

# Итерация списка с использованием `for + enumerate`

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

- Вывод:

`k=0 v=10`

`k=1 v=S`

`k=2 v=15`

`k=3 v=A`

`k=4 v=1`



## 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:

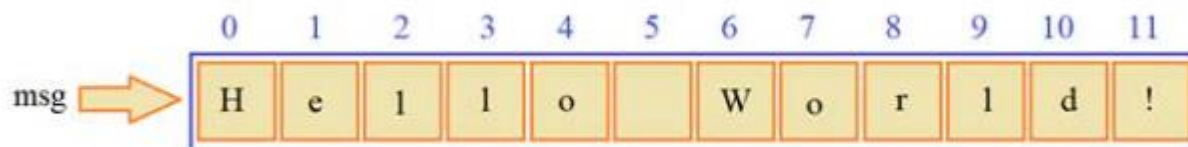
`[[1,2,3], [10, 20], [100]] -> 136`



# 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}                   |

# Индексация строки (вспомним)



Можно использовать и отрицательные индексы!

1. Что напечатает эта программа?

```
msg = 'Hello world!'
```

```
for k in range(-12, 12):
```

```
    print(msg[i], end = "")
```

2. А что напечатает `print(msg[::-1])`?



# Некоторые функции по работе со строками

`s = 'Abra cad abra'`

Проверка вхождения подстроки

```
print('ad' in s)
```

Поиск (возвращается индекс или -1):

```
print(s.find('cad'))
```

Длина:

```
print(len(s))
```

Количество вхождений подстроки

```
print(s.count('br'))
```

Различные преобразования, например:

```
s.upper(), s.lower()
```

Различные проверки:

```
s.islower(), '123'.isdigit()
```

Преобразование чего угодно в строки:

```
str(123), str(True), str([1,2,3])
```

Разбиение строки на элементы

```
s.split()
```

и многие другие.

```
dir(str)
```

Давайте выполним эти функции





## Задание


**Написать программу, которая определяет, является ли строка палиндромом**

Ввод: строка (например: abссба, или aaabbbcbbaaa, или хузхуз)

Вывод:

**True** – если строка является палиндромом (т.е. читается одинаково слева направо и справа налево, например, «казак», «мадам»)

**False** – если строка не является палиндромом.



## Очень полезная функция – join (`str.join(iterable)`)

Допустим у нас есть список из слов, например:

```
lst = ['Я', 'пишу', 'программы', 'на', 'Питоне']
```

Что напечатает <code>print(' '.join(lst))</code> ?	# один пробел
Что напечатает <code>print('').join(lst)</code> ?	# без пробела
Что напечатает <code>print(', '.join(lst))</code> ?	# запятая и пробел
Что напечатает <code>print('\n'.join(lst))</code> ?	# перенос строки



## «Обратная» к ней функция `split()`

**`str.split()`** (*sep=None, maxsplit=-1*)

Разбиение строки `str`, используя разделитель `sep`.

Что напечатает:

```
print("Don't worry be happy".split())
```

```
print("Don't  worry  be  happy".split())
```

```
print("Don't  worry  be  happy".split(' '))
```

```
print("Don't  worry  be  happy".split(maxsplit=1))
```



# 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 list(iterable)

Что напечатает:

```
print(list('abcdef'))
```

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

```
print(list([1,2,3]))
```

```
print(list((1,2,3)))
```

```
print(list({1,2,3}))
```



## Элементы списка разных типов. Индексы

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

Напечатайте элементы списка `lst`  
от индекса `-len(lst)` до индекса `len(lst)`

list


- append()
- clear()
- copy()
- count()
- extend()
- index()
- insert()
- pop()
- remove()
- reverse()
- sort()

# Python List Methods



WTMatter





## Метод `append()`

`(s.append(x))`

Дан пустой список `lst = []`

Введите число `n = int(input())`

Используя метод `append()`, сгенерируйте список:

`lst = [1, 2, 2, 3, 3, 3, 4, 4, 4, 4, ..., n, n, ..., n, n]`



# Функции по работе со списками

Operation	Result	Результат
<code>s[i] = x</code>	item <code>i</code> of <code>s</code> is replaced by <code>x</code>	<code>i</code> – ый элемент <code>s</code> меняется на <code>x</code>
<code>s[i:j] = t</code>	slice of <code>s</code> from <code>i</code> to <code>j</code> is replaced by the contents of the iterable <code>t</code>	Срез от <code>i</code> до <code>j</code> меняется на содержимое <code>t</code> (iterable)
<code>del s[i:j]</code>	same as <code>s[i:j] = []</code>	Удалить срез списка
<code>s[i:j:k] = t</code>	the elements of <code>s[i:j:k]</code> are replaced by those of <code>t</code>	Срез с шагом <code>k</code> меняется на содержимое <code>t</code>
<code>del s[i:j:k]</code>	removes the elements of <code>s[i:j:k]</code> from the list	Срез с шагом удаляется
<code>s.append(x)</code>	appends <code>x</code> to the end of the sequence (same as <code>s[len(s):len(s)] = [x]</code> )	Добавить <code>x</code> к концу списка
<code>s.clear()</code>	removes all items from <code>s</code> (same as <code>del s[:]</code> )	Удалить все значения списка
<code>s.copy()</code>	creates a shallow copy of <code>s</code> (same as <code>s[:]</code> )	Копия списка
<code>s.extend(t)</code> or <code>s += t</code>	extends <code>s</code> with the contents of <code>t</code> (for the most part the same as <code>s[len(s):len(s)] = t</code> )	Добавления к списку содержимого списка <code>t</code>
<code>s *= n</code>	updates <code>s</code> with its contents repeated <code>n</code> times	Содержание списка повторяется <code>n</code> раз
<code>s.insert(i, x)</code>	inserts <code>x</code> into <code>s</code> at the index given by <code>i</code> (same as <code>s[i:i] = [x]</code> )	Вставляет <code>x</code> на <code>i</code> -ое место
<code>s.pop()</code> or <code>s.pop(i)</code>	retrieves the item at <code>i</code> and also removes it from <code>s</code>	Выдает значение и удаляет его из <code>s</code>
<code>s.remove(x)</code>	remove the first item from <code>s</code> where <code>s[i]</code> is equal to <code>x</code>	Удаляет первое вхождение <code>x</code> в <code>s</code>
<code>s.reverse()</code>	reverses the items of <code>s</code> in place	Переворачивает список



## Max, min, sum

```
lst = [1, -2, 3, -4, 5, -6, 7, -8, 9, -10]
```

Что напечатает:

```
print(max(lst), min(lst), sum(lst))
```

Можно использовать параметр `key = функция`

Например:

```
print(min(lst, key = abs))
```



## Метод count(x)

Возвращает количество вхождений элемента **x** в список.

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

```
>>> print(a.count(2))
```

```
2
```

А что вернет, если в списке нет этого элемента?



## Метод `index(x[, start[, end]])`

Возвращает индекс элемента.

```
>>> a = [1, 2, 3, 4, 5]
```

```
>>> a.index(4)
```

```
3
```

А что вернет, если элемента нет в списке?



## Методы `sorted()` и `sort()`

Сортируют элементы в списке по возрастанию.

Для сортировки в обратном порядке используйте флаг `reverse=True`.

Дополнительные возможности открывает параметр `key = функция`

```
a = [1, 4, -2, -8, 1]
```

```
a.sort()
```

```
print(a)
```

```
a.sort(key = abs)
```

```
print(a)
```

```
b = sorted(a)
```

```
print(a)
```

```
print(b)
```

[illegible]



# Цикл while

Общая конструкция:

**while** проверка условия:

операторы

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



## Пример

`i = 5`

`while i <= 15:`

`print(i)`

`i = i + 2`

5

7

9

11

13

15

# А если поменять местами два оператора в теле цикла?





# Бесконечный цикл

```
i = 5
```

```
while True:
```

```
    print(i)
```

```
    i = i + 2
```

```
    if i == 9: break
```

**Что выведет код ?**



## Задание

На вход подается последовательность целых чисел.

Окончанием ввода является ввод отрицательного числа.

Напечатайте сумму всех чисел, кроме этого отрицательного числа.



# 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}



# Кортежи (tuple)

Они являются упорядоченными коллекциями произвольных объектов

Поддержка доступа по индексу

Неизменяемые последовательности

Имеют фиксированную длину

Представляют из себя массив ссылок на объекты



## Как получить кортеж?

Напечатайте:

```
print(tuple([1,2,3]))
```

```
print(tuple('abcd'))
```

```
print(tuple((1,)))
```

```
a,b,c = 1, 'a', True
```

```
tpl = (a,b,c)
```

```
print(tpl[0], tpl[1], tpl[2])
```

Попробуйте сделать `tpl[0] = 123`, что произойдет?



# Множественное присваивание

```
x, y = 100, 200
```

```
( x, y ) = (100, 200)
```

```
print(x)
```

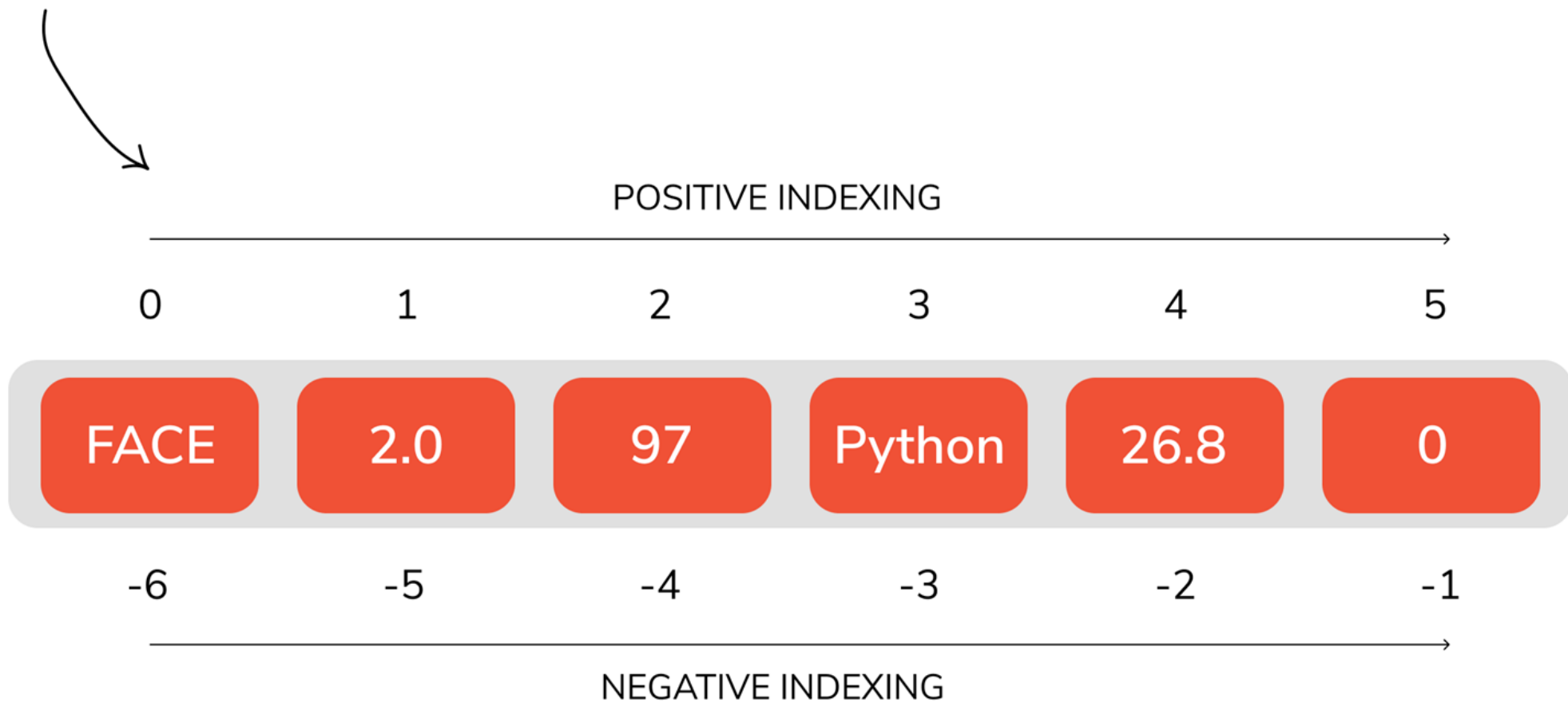
```
100
```

```
print(y)
```

```
200
```

# Индексация

Tuple = ('FACE', 2.0, 97, 'Python', 26.8, 0 )



# Python Tuple Methods

tuple.  
    → count()  
    → index()







# Индекс заданного элемента

## **index(value, start, stop)**

```
rom = ('I', 'II', 'III', 'IV', 'V', 'VI', 'VII', 'VIII', 'IX', 'X')
```

```
print(rom.index('X'))
```

9

```
str = ('aa', 'bb', 'aa', 'cc')
```

```
print(str.index('aa'))
```


0

```
str = ('aa', 'bb', 'aa', 'cc' )
```

```
print(str.index('aa', 1, len(str)))
```

```
print(str.index('aa', 1,))
```

2



## Число вхождений элемента **count()**

```
t_str = ('aa', 'bb', 'aa', 'cc')
```

```
print(t_str.count('aa'))
```

```
2
```



## Задание

Дан кортеж (123, 234, 345, 456, 567, 678, 789, 890).

Вводится еще одно целое число больше 0.

Создайте новый кортеж из первого кортежа и введенного числа, чтобы в новом кортеже числа не убывали.



## Об одном свойстве кортежей

```
tpl = (1, 2, 3, [11, 22, 33])
```

```
tpl[3].append(44)
```

```
print(tpl)
```

Что будет напечатано?

# Общие функции для list, tuple

Некоторые из них работают для str и range

Operation	Result	Результат
<code>x in s</code>	True if an item of <code>s</code> is equal to <code>x</code> , else False	<b>True</b> , если какой-то элемент <code>s</code> равен <code>x</code> , иначе <b>False</b>
<code>x not in s</code>	False if an item of <code>s</code> is equal to <code>x</code> , else True	Наоборот
<code>s + t</code>	the concatenation of <code>s</code> and <code>t</code>	Конкатенация <code>s</code> и <code>t</code>
<code>s * n</code> (or <code>n * s</code> )	equivalent to adding <code>s</code> to itself <code>n</code> times	Эквивалентно сложению с собой <code>n-1</code> раз
<code>s[i]</code>	<code>i</code> th item of <code>s</code> , origin 0	<code>i</code> – ый элемент, начиная с 0
<code>s[i:j]</code>	slice of <code>s</code> from <code>i</code> to <code>j</code>	Срез от <code>i</code> до <code>j</code> неключительно
<code>s[i:j:k]</code>	slice of <code>s</code> from <code>i</code> to <code>j</code> with step <code>k</code>	Срез с шагом <code>k</code>
<code>len(s)</code>	length of <code>s</code>	Длина
<code>min(s)</code>	smallest item of <code>s</code>	Наименьший элемент
<code>max(s)</code>	largest item of <code>s</code>	Наибольший
<code>s.index(x[, i[, j]])</code>	index of the first occurrence of <code>x</code> in <code>s</code> (at or after index <code>i</code> and before index <code>j</code> )	Индекс первого вхождения <code>x</code> в <code>s</code> , начиная с <code>i</code> до <code>j</code> неключительно
<code>s.count(x)</code>	total number of occurrences of <code>x</code> in <code>s</code>	Количество вхождений <code>x</code> в <code>s</code>



## Задача 3-1

Вводить в бесконечном цикле зарплаты сотрудников.

Окончание ввода – ввод 0.

После чего напечатать среднюю зарплату.



## Задача 3-2

Дано целое число.

Сосчитать и напечатать, сколько в его записи нулей, единиц, двоек и т.д.

Например:

Ввод: 133244459

Вывод:

0 - 0

1 – 1

2 – 1

3 – 2

4 – 3

5 – 1

6 - 0

и т.д.



## Задача 3-3

На вход подается предложение из нескольких слов.

Слова разделены пробелами.

Напечатайте первое самое длинное слово в этом предложении.

Более сложный вариант, напечатать все самые длинные слова,  
если их несколько с наибольшей длиной