

# PyBursa #6

Webinar 02 (19:00 21.02.2017)

# Webinar Roadmap

- Ваш Рейтинг. Каким образом идет расчет: 70% + 30%
- Примеры решения задач КР. Тесты
- Неделя 2:
  - *на что обратить внимание*
  - *система проверки заданий Недели № 2*
- Предложение:
  - *вопросы на вебинар – направить ассистенту*

# Ваш Рейтинг

**1.Казнадий Роман**  
**2.Малявин Михаил**  
**3.Добродий Игорь**  
4.Бурлаков Алексей  
5.Яценко Максим  
6.Mazur Artem

1.roman0001  
2.superduper  
3.dobrodiy  
4.Alex\_Tsumibito  
5.maxyko  
6.Mazart

Рейтинг



## Практическое задание №1\_6 #2

(1/1 point)

Какие пояснения следующего кода правильны:

```
a = 'julie'  
b = 'JULIE'  
c = 'Julie'
```

```
if a == c or b:  
    print(True)  
else:  
    print(False)
```

Результат - значение переменной b, а поскольку она не пустая, то всегда будет True вне зависимости от регистра

☒ проверка `a == c` не будет выдержана (сама по себе вернет `False`), т.к. при сравнении строк учитывается регистр: заглавные буквы имеют код ASCII таблицы, отличный от кода строчных букв;

☐ в результате исполнения кода получим `False`, потому что `a` не равно `c` и `a` не равно `b`, ведь учитывается регистр;

☒ в любом случае (при любом регистре значимых значений `a`, `b` и `c`) в результате выполнения кода получим `True`;

☐ только при преобразовании `c.lower()` или `b.lower()` до запуска оператора `if` в



## Практическое задание №1\_4 #2

(1/1 point)

Какие варианты кода вернут значение по ключу 'version' из словаря

```
dictD = {'version': '2.7.10', 'language': 'python'} :
```

`dictD['version']``dictD[0]``dictD.pop('version')``dictD.popitem()``list(dictD.values())[0]`

1) Не возвращает значение по ключу

`dictD.get('version')`

2) Словарь неупорядоченный тип данных, а список - Да

Результат - не предсказуем!!!





## КР 1.3

### ЗАДАНИЕ 3 (10 баллов из 10)

Подсчет гласных букв в строке `var_string`.

Примечание:

- для простоты строка `var_string` состоит из букв латинского алфавита;
- набор гласных принимаем за 'a', 'e', 'i', 'o', 'u';
- программа должна быть нечувствительна к регистру.

```
counting_vowels = 0
for char in var_string:
    if char.lower() in ['a', 'e', 'i', 'o', 'u']:
        counting_vowels += 1
print(counting_vowels)
```

Решения идентичны, но со строкой  
перечислить гласные проще

```
vowels = 'aeiou'
s = var_string.lower()
count_vowels = 0
for i in s:
    if i in vowels:
        count_vowels += 1
print(count_vowels)
```

## КР 1.4 Поиск подстроки 'wow'

### ЗАДАНИЕ 4 (10 баллов из 10)

Реализовать подсчет количества вхождений подстроки "wow" в строке `var_string`.

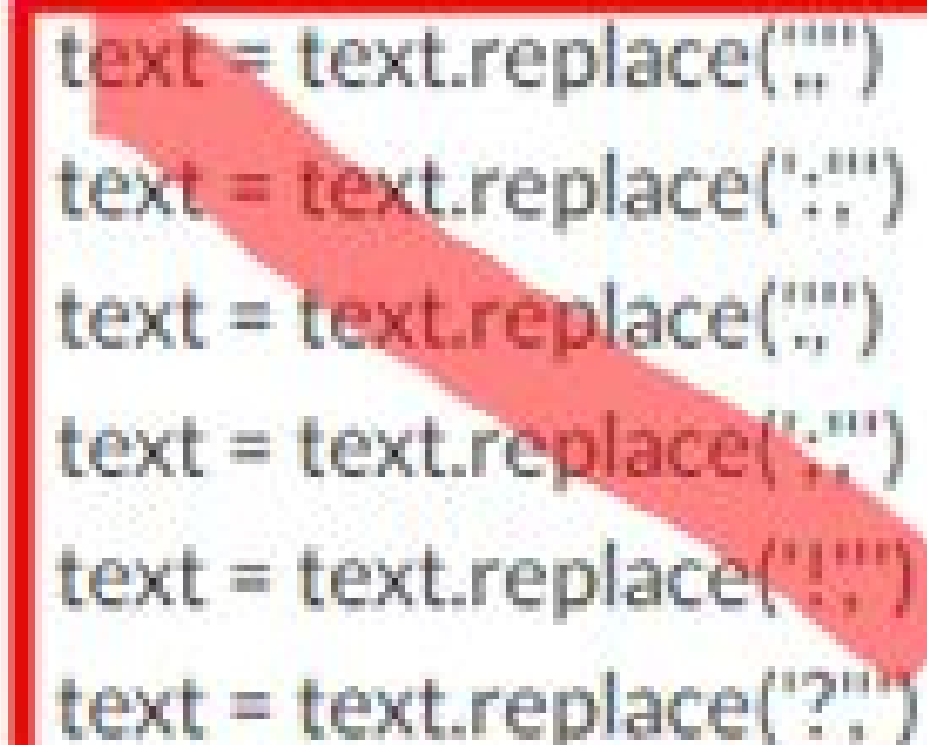
### Использование срезов *via find*

```
count = 0
for i in range(len(var_string)):
    if var_string[i:i+3] == 'wow':
        count += 1
print(count)
```

```
count = 0
i = 0
while i < len(var_string):
    if var_string.find('wow', i) == -1:
        break
    else:
        i = var_string.find('wow', i)
        i += 2
        count += 1
```

!!! Правильно использовать вспомогательные переменные: **pattern = 'wow'** !!!

## КР 2.1 Задача с палиндромами. Приведение строки к рабочему виду



```
text = text.replace(",")
text = text.replace(":",")
text = text.replace(".",")
text = text.replace(";")
text = text.replace("!")
text = text.replace("?")
```

Много готовых решений для работы со строками.  
Можно выстраивать цепочкой:

```
text = 'sos, a b tr!!!'
palindromes = 0
for word in text.split():
    word = word.lower().rstrip(',.:;!?')
    if word == word[::-1]:
        palindromes += 1
print(palindromes)
```

Перебрали по  
символам –  
сложили заново:

```
signs = ',.:;!?'
text = text.lower()
text_new = ''

for i in text:
    if i not in signs:
        text_new += i
```



## КР 2.2 На что обратить внимание

### ЗАДАНИЕ 2 (10 баллов из 10)

Написать фрагмент python кода, который модифицирует строку чисел `var_string` следующим образом:

- в начале строки идут нечетные числа в порядке возрастания
  - далее идут четные числа в порядке убывания
- и выводит на печать измененную строку `var_string` .

- Задача требовала внимательного обращения с типами данных
- Понимания работы цикла и условного оператора
- Преобразования списков в строки и строк в списки
- Функция `join()`

## КР 2.2 Варианты: со строками и списками

```
var_string = '2'
change_string = ''
x=''
y=''
for i in var_string:
    if int(i)%2==0:
        x += i
    else:
        y += i
y1=sorted(y)
x1=sorted(x)
x1=''.join(x1)
y1=''.join(y1)
#print (y1,x1)
change_string=y1+x1[::-1]
print(change_string)
```

Цикл закончил

```
var_string = "7823947382"
var_string = list(var_string)
division_by_two = []
not_division_by_two = []
for item in var_string:
    if int(item)%2 == 0:
        division_by_two.append(item)
    if int(item)%2 != 0:
        not_division_by_two.append(item)
l2 = not_division_by_two.sort()
l1 = division_by_two.sort()
l1 = ''.join([str(i) for i in division_by_two])
l2 = ''.join([str(i) for i in not_division_by_two])
change_string = l2 + l1[::-1]
print(change_string)
```

Следим за типом данных

можно с else

## КР 2.2 Вариант: использован материал из недели 2

```
var_string='2738462384'  
odds = filter(lambda x: int(x) % 2 == 1, list(var_string))  
evens = filter(lambda x: int(x) % 2 == 0, list(var_string))  
change_string = ''.join(sorted(odds) + sorted(evens, reverse=True))  
print( change_string)
```

## КР 2.3 Поиск самой длинной подстроки

Требуются пояснения по ходу решения задачи

```
longest = ''

if var_string:
    curString = var_string[0]
    longest = var_string[0]

    for i in range(1, len(var_string)):


        if var_string[i] >= curString[-1]:
            curString += var_string[i]
            if len(curString) > len(longest):
                longest = curString
        else:
            curString = var_string[i]
print(longest)
```

# WEEK 2. Функции: на что обратить внимание

- **Декларирование** функции — это **определение** функции;  
**вызов** — это использование ранее определенной (декларированной) функции
- **None** — дефолтное возвращаемое значение (если не прописано иначе)

Шаг 5 из 5, функция sum **запустить**

```
1 def sum(a,b):  
2     print(a + b)  
3  
4 sum(3,2)  
5
```



sum

a	3
b	2

Функция собирается  
завершиться

Возвращаемое  
значение:

None



# Функции: важно!

- **return** — выход из функции  
(в примере до *print* не дойдет):

```
def bigger(a,b):  
    if a > b:  
        return a  
    else:  
        return b  
    print(a,b)
```

- Значение, которое возвращает функция зависит от нескольких факторов:

```
def sum(a,b):  
    return a + b
```

```
s1 = sum(3,2)  
s2 = sum('3','2')
```

- Правильнее один **return** :

```
def smaller(a,b):  
    if a < b:  
        return a  
    else:  
        return b
```

# Правильнее один return:

```
def smaller(a,b):  
    if a < b:  
        sm = a  
    else:  
        sm = b  
    return sm
```

Глобальные  
переменные

s1	5
s2	"32"

# Списки как параметры Функции: на что обратить внимание 1:

➤ <http://pythontutor.ru/visualizer/saved/2de66d5fa9/>

Шаг 5 из 8, функция foo **запустить** ☐ выполнить пошагово ☒

```

1 numbers_list = [3, 6, 1, 8]
2
3 def foo(ls):
4     return ls.sort()
5
6 print('Before:', numbers_list)
7 print('Call:', foo(numbers_list))
8 print('After:', numbers_list)
9

```

К началу | Назад | Далее | К концу

Стек

растет

ВНИЗ ▼

Глобальные  
переменные

numbers\_list

foo

foo

ls

list (id=1):

0	1	2	3
3	6	1	8

function instance (id=2)



# Функции. Значения по умолчанию для аргументов вычисляются только один раз

Прекрасное объяснение [здесь](#) см.5.1 (или [здесь](#))

➤ <http://pythontutor.ru/visualizer/saved/98e7195c77/>

Шаг 12 из 13, функция foo **запустить** ☒ выполнить пошагово

```

1 def foo (numbers_list = []):
2     if numbers_list:
3         numbers_list.append(1)
4     else:
5         numbers_list.append(2)
6     return numbers_list
7
8 foo()
9 foo()
```

Стек

растет

ВНИЗ ▼

[Глобальные переменные](#)

foo

[foo](#)

numbers\_list

function instance (id=1)

list (id=2):

0	1
2	1

# Проверка на тип данных:

`isinstance` ИЛИ `type`

➤ Какую функцию лучше выбрать:

(<http://stackoverflow.com/questions/1549801/differences-between-isinstance-and-type-in-python>):

Using `type()` :

```
import types

if type(a) is types.DictType:
    do_something()
if type(b) in types.StringTypes:
    do_something_else()
```

Using `isinstance()` :

```
if isinstance(a, dict):
    do_something()
if isinstance(b, str) or isinstance(b, unicode):
    do_something_else()
```

➤ В code review будет использован `isinstance`



# Обработка ошибок

## Конструкция try ... except ...

➤ Смотрите материалы лекции Недели 1:

<b>try,</b> <b>except</b>	<p>except <u>IndexError</u>:</p> <p>(не перехватит ошибку <u>нейма</u> и другие ошибки; перехватит только ошибку индекса); в нашем примере ошибка возникает, если не введено необходимое значение для <u>sys.argv</u>. Пример «перехвата»:</p> <pre>try:     r = sys.argv[1] except IndexError:     print("Error!")     exit(1)</pre> <p>В Python реализован «перехват»:</p> <p><u>ValueError</u></p> <p><u>NameError</u></p> <p><u>KeyError</u></p> <p><u>SyntaxError</u></p> <p><u>ZeroDivisionError</u></p>	<p>Предназначение: локализовать точки программы, в которых могут возникнуть ошибки, и перехватить эти ошибки (и сразу совет: не нужно помещать в блок <u>трай-ексепт</u> большой блок кода).</p> <pre>import sys import math  try:     r = sys.argv[1] except IndexError:     print("Error!")  if r.isdigit():     r = int(r) else:     print("radius is not number!")     exit(1)  s = math.pi * r**2 print("Square: %d" %s)</pre>
------------------------------	--	---



# Полезные Built-in Functions

➤ len, range, min, max...

➤ <https://docs.python.org/3/library/functions.html>

## 2. Built-in Functions

The Python interpreter has a number of functions and types built into it that are always available. The order.

Built-in Functions				
abs()	dict()	help()	min()	setattr()
all()	dir()	hex()	next()	slice()
any()	divmod()	id()	object()	sorted()
ascii()	enumerate()	input()	oct()	staticmethod()
bin()	eval()	int()	open()	str()
bool()	exec()	isinstance()	ord()	sum()
bytearray()	filter()	issubclass()	pow()	super()
bytes()	float()	iter()	print()	tuple()
callable()	format()	len()	property()	type()
chr()	frozenset()	list()	range()	vars()
classmethod()	getattr()	locals()	repr()	zip()
compile()	globals()	map()	reversed()	__import__()
complex()	hasattr()	max()	round()	
delattr()	hash()	memoryview()	set()	

# Пробелы и табуляция

- Не смешивать!!!
- Ошибка:
  - inconsistent use of tabs and spaces in indentation

# Система проверки заданий Недели № 2

- **Не ставить принты** для КЗ Недели № 2
  - это исходит из постановки задачи: «написать функцию, которая **возвращает** результат»!
- Необходимо только **декларирование** функции, которая возвращает значение.
- Иначе **сгорает** попытка и выдается лог:
  - *"Your submission could not be graded. Please recheck your submission and try again. If the problem persists, please notify the course staff."*

# Ваши вопросы?