

Лекция 2

«Основы программирования на Python»



К.т.н. доц. Ляшенко Алексей Сергеевич



Содержание

- **Кортежи (tuple)**
- **Множества (set и frozenset)**
- **Инструкции**
- **Циклы**
- **Функции**



Кортежи (tuple)

- Кортежи – неизменяемый список

```
>>> x = tuple()
```

```
>>> x=()
```

```
>>> x='stop',
```

Пример

```
>>> x=(23,34, 5, 8,-1)
```

```
>>> z={(2,1,1) : 23}
```

```
>>> z
```

```
{(2, 1, 1): 23}
```



Множество

- Множество == контейнер, с не повторяющимися элементами

```
>>> m = set()
```

```
>>> m
```

```
set()
```

```
>>> m = set('Студент')
```

```
>>> m
```

```
{'н', 'т', 'д', 'е', 'с', 'у'}
```



Генератор множеств

```
>>> m = {a**3 for a in range(7)}
```

```
>>> m
```

```
{0, 1, 64, 8, 216, 27, 125}
```

Множество можно использовать для
удаления повторных элементов

```
>>> tag = ['h1', 'h2', 'font', 'font', 'h1']
```

```
>>> set(tag)
```

```
{'h1', 'h2', 'font'}
```



Операции над множествами

- `len(s)` - число элементов в множестве (размер множества).
- `x in s` - принадлежит ли `x` множеству `s`.
- `set.isdisjoint(other)` - истина, если `set` и `other` не имеют общих элементов.
- `set == other` - все элементы `set` принадлежат `other`, все элементы `other` принадлежат `set`.
- `set.issubset(other)` или `set <= other` - все элементы `set` принадлежат `other`.
- `set.issuperset(other)` или `set >= other` - аналогично.
- `set.union(other, ...)` или `set | other | ...` - объединение нескольких множеств.
- `set.intersection(other, ...)` или `set & other & ...` - пересечение.
- `set.difference(other, ...)` или `set - other - ...` - множество из всех элементов `set`, не принадлежащие ни одному из `other`.
- `set.symmetric_difference(other)`; `set ^ other` - множество из элементов, встречающихся в одном множестве, но не встречающихся в обоих.
- `set.copy()` - копия множества.



Операции над множествами

- **set.update**(other, ...); set |= other | ... - объединение.
- **set.intersection_update**(other, ...); set &= other & ... - пересечение.
- **set.difference_update**(other, ...); set -= other | ... - вычитание.
- **set.symmetric_difference_update**(other); set ^= other - множество из элементов, встречающихся в одном множестве, но не встречающиеся в обоих.
- **set.add**(elem) - добавляет элемент в множество.
- **set.remove**(elem) - удаляет элемент из множества. `KeyError`, если такого элемента не существует.
- **set.discard**(elem) - удаляет элемент, если он находится в множестве.
- **set.pop**() - удаляет первый элемент из множества. Так как множества не упорядочены, нельзя точно сказать, какой элемент будет первым.
- **set.clear**() - очистка множества.



frozenset

```
>>> x=set('Слово')
>>> y=frozenset('Слово')
>>> x.add(2)
>>> y.add(2)
```

Traceback (most recent call last):

```
File "<pyshell#23>", line 1, in <module>
    y.add(2)
```

```
AttributeError: 'frozenset' object has no attribute
'add'
```




Инструкции if

Общая форма записи условной инструкции `if` выглядит следующим образом:

```
if test1:
```

```
    state1
```

```
elif test2:
```

```
    state2
```

```
else:
```

```
    state3
```

```
if 0:
```

```
    print('правда')
```

```
else:
```

```
    print('ложь')
```

```
ложь
```



Проверка истинности в Python

- Любое число, не равное 0, или непустой объект - истина.
- Числа, равные 0, пустые объекты и значение None - ложь
- Операции сравнения применяются к структурам данных рекурсивно
- Операции сравнения возвращают True или False
- Логические операторы and и or возвращают истинный или ложный объект-операнд

Трехместное выражение if/else

```
>>> A = 'истина' if 'Слово' else 'ложь'
```

```
>>> A
```

```
истина
```



Цикл while

```
>>> i=10
```

```
>>> while i<=14:
```

```
    print(i)
```

```
    i=i+2
```

```
10
```

```
12
```

```
14
```



Цикл for

```
>>> for x in 'Привет Универ':  
    print(x*2, end="")
```

ППррииввеетт УУннииввеерр



Оператор continue

```
>>> for x in 'привет Универ':  
    if x == 'и':  
        continue  
    print(x*3, end="")
```

ппррррвввеееттт уууунннвввеееррр



Оператор break

```
>>> for x in 'привет Универ':  
    if x == 'и':  
        break  
    print(x*3, end="")
```

пппрррр



else

```
>>> for x in 'привет Универ':  
    if x == 'g':  
        break  
    else:  
        print('Данного элемента нет')
```

Данного элемента нет



Функции

- Функция == объект, который принимает аргументы и возвращает значение

```
>>> def z(x,y):  
    return x+y
```

```
>>> z(2,7)
```

```
9
```

```
>>> z('Слово 1', ' Слово 2')
```

```
'Слово 1 Слово 2'
```



Функции в файле index.py

```
def say():  
    print('Привет, Универ!') # блок,  
    принадлежащий функции  
# Конец функции  
say() # вызов функции  
say() # ещё один вызов функции
```

```
Привет, Универ!  
Привет, Универ!
```



Функция в функции

```
>>> def nFunc(x):  
        def mFunc(y):  
            return x*y  
        return mFunc
```

```
>>> z = nFunc(10)
```

```
>>> z(15)
```

```
150
```

Значения аргументов по умолчанию

Часть параметров функций могут быть *необязательными*, и для них будут использоваться некоторые заданные значения по умолчанию, если пользователь не укажет собственных. Этого можно достичь с помощью значений аргументов по умолчанию. Их можно указать, добавив к имени параметра в определении функции оператор присваивания (=) с последующим значением.

```
def say(message, times = 1):  
    print(message * times)
```

```
say('Привет')  
say('Универ', 5)
```

```
Привет  
УниверУниверУниверУниверУнивер
```

Ключевые аргументы функции

```
def func(a, b=5, c=10):  
    print('а равно', a, ', б равно', b, ', а с  
равно', c)
```

```
func(3, 7)
```

```
func(25, c=24)
```

```
func(c=50, a=100)
```

а равно 3, б равно 7, а с равно 10

а равно 25, б равно 5, а с равно 24

а равно 100, б равно 5, а с равно 50



Локальные переменные

```
x = 50
```

```
def func(x):
```

```
    print('x равен', x)
```

```
    x = 2
```

```
    print('Замена локального x на', x)
```

```
func(x)
```

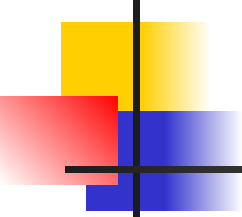
```
print('x по-прежнему', x)
```

x равен 50

Замена локального x на 2

x по-прежнему 50

Зарезервированное слово "global"



```
x = 50
```

```
def func():  
    global x
```

```
    print('x равно', x)
```

```
    x = 2
```

```
    print('Заменяем глобальное значение x на', x)
```

```
func()
```

```
print('Значение x составляет', x)
```

x равно 50

Заменяем глобальное
значение x на 2

Значение x составляет 2

Зарезервированное слово "nonlocal"

```
def func_outer():  
    x = 2  
    print('x равно', x)  
    def func_inner():  
        nonlocal x  
        x = 5  
        func_inner()  
        print('Локальное x сменилось на', x)
```

func_outer()

x равно 2

Локальное x сменилось на 5



Оператор "return"

Оператор `return` используется для возврата из функции, т.е. для прекращения её работы и выхода из неё. При этом можно также вернуть некоторое значение из функции.

```
def maximum(x, y):  
    if x > y:  
        return x  
    elif x == y:  
        return 'Числа равны.'  
    else:  
        return y  
print(maximum(2, 3))
```