ЛЕКЦИЯ №5 ПО ТЕХНОЛОГИИ ПРОГРАММИРОВАНИЯ

Москва, 2020

Типы доступа к файлам

Произвольный доступ (Открытие как бинарного)

Последовательный доступ к файлу (Открытие как текстового)

```
jpeg = [120, 3, 255, 0, 100]
with open(r'C:\tmp\btest.bin', 'w+b') as file:
    file.write(bytes(jpeg))
```

```
with open(r'C:\tmp\btest.bin', 'rb') as file:
    data = file.read()
    for b in data:
        print(int(b))
```

Типы доступа к файлам

```
jpeg = [120, 3, 255, 0, 100]
with open(r'C:\tmp\btest.bin', 'w+b') as file:
    file.write(bytes(jpeg))
```

```
with open(r'C:\tmp\btest.bin', 'rb') as file:
    data = file.read()
    for b in data:
        print(int(b))
```

Типы данных

Имя	Тип	Описание
Целые числа	int	1, 2, 3
Числа с плавающей точкой	float	1.1, 2.4, 3.0
Строки	str	последовательность символов: "Hello", 'hello', "42"
Списки	list	последовательность объектов: `[1, 2.0, "hello"]`
Словари	dict	Список пар «ключ-значение»: `{"john" : "+1-23-45", "bob" : "+1-32-65"}`
Кортежи	tuple	Последовательность неизменяемых объектов: `(1, 2.0, "hello")`
Множества	set	последовательность уникальных объектов: `{"a", "b"}`
Булевые значения	bool	логические значения: True или False

Списки

Объединение списков

Добавление в конец списка: append()

Извлечение последнего элемента: pop()

Сортировка: sort(reverse = False)

Сравнение строк в лексикографическом порядке

```
letters = ['abc','a', 'bc']

letters.sort(key=len)
```

Вывод в обратном порядке: reverse()

Добавление в любую позицию списка: letters.insert(1,'d')

Возвращает индекс фиксированного элемента: letters.index('d')

Количество элементов в списке: letters.count(‘d’)

Массив

```
n = 3
m = 2
a = []
for i in range(m):
    row = [0.0]*n
    a+=[row]
```

```
a = []
for i in range(n):
    a+=[0.0]
```

Другая особенность языка в том, что он предоставляет возможность умножать list на число и результатом этой операции будет list в котором все элементы скопированы указанное количество раз. Давайте я покажу вам несколько примеров:

```
>>> [1] * 3
[1, 1, 1]
>>> [1] * 5
[1, 1, 1, 1, 1]
>>> [1] * 0
[]
>>> [1, 2] * 3
[1, 2, 1, 2, 1, 2]
```

Словари

Список пар: ключ, значение

Создаются через фигурные скобки и конструкцию dict

Обращаться по ключу или с помощью get: dict1.get('Ivanov')

Получить список ключей: list(dict1.keys())

Проверить, что ключ в словаре: ‘Ivanov’ in dict1 , not in

Получить значений словаря: dict1.values()

Обход словаря:

```
for k,v in dict1.items():
    print(k,v)
```

Удаление из списка по ключу:

```
dict1.pop('Sidorov')
```

Упорядоченные словари

```
from collections import OrderedDict
```

```
d1 = OrderedDict()
d1['a'] = 'A'
d1['b'] = 'B'
d1['c'] = 'C'

d2 = OrderedDict()
d2['b'] = 'B'
d2['a'] = 'A'
d2['c'] = 'C'

d3 = OrderedDict()
d3['a'] = 'A'
d3['b'] = 'B'
d3['c'] = 'C'
```

```
print(d1==d2)
print(d1==d3)
```

```
False
True
```

Кортежи

Неизменяемые списки (turple)

```
strings = ('str1', 'str2', 'str3')
```

Person = (‘Ivan’, ‘Ivanov’, 22)

Person.count(‘Ivan’)

Person.index(‘Ivan’)

```
persons = [('John', 22), ('Bob', 32), ('Dave', 20)]
len(persons)
```

```
3
```

```
for (name, age) in persons:
    print(f'{name} is {age} years old')
```

```
John is 22 years old
Bob is 32 years old
Dave is 20 years old
```

Именнованные кортежи

```
from collections import namedtuple
```

```
Player = namedtuple('Player', 'name age rating')
```

```
players = [Player('Carlsen', 1990, 2842), Player('Caruana', 1992, 2822), Player('Mamedyarov', 1985, 280
```

```
players[0]
```

```
Player(name='Carlsen', age=1990, rating=2842)
```

```
players[0].name
```

```
'Carlsen'
```

Множества

```
my_set = set()
print(my_set)
print(type(my_set))
```

```
set()
<class 'set'>
```

```
my_set.add(1)
my_set
```

```
{1}
```

```
my_set.add(2)
my_set
```

```
{1, 2}
```

```
my_set.add(2)
my_set
```

```
{1, 2}
```

```
my_list = [1,1,1,1,2,2,2,2,3,3,3,4]
s = set(my_list)
s
```

```
{1, 2, 3, 4}
```

```
set1 = {1,2,3,4}
set2 = {1,2,3,4,5}
```

```
set1.issubset(set2)
```

```
True
```

```
set2.issubset(set1)
```

```
False
```

Циклы внутри последовательности

```
greeting = 'hello, world'
chars = []
for l in greeting:
    chars.append(l)
chars
```

```
['h', 'e', 'l', 'l', 'o', ',', ' ', 'w', 'o', 'r', 'l', 'd']
```

```
chars = [l for l in greeting]
chars
```

```
['h', 'e', 'l', 'l', 'o', ',', ' ', 'w', 'o', 'r', 'l', 'd']
```

```
chars = [l for l in 'bla-bla-bla']
chars
```

```
['b', 'l', 'a', '-', 'b', 'l', 'a', '-', 'b', 'l', 'a']
```

```
numbers = [0,1,2,3,4,5,6,7,8,9,10]
numbers
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

```
numbers = [n for n in range(0, 11)]
numbers
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

Циклы внутри последовательности

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

```
numbers  = [n*n for n in range(0,11)]
numbers
```

```
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

```
numbers = [n*n for n in range(0,11) if n%2!=0]
numbers
```

```
[1, 9, 25, 49, 81]
```

```
ratings = [2485, 2580, 2480, 2600, 2482, 2520]
titles = ['GM' if x>=2500 else 'MM' for x in ratings]
titles
```

```
['MM', 'GM', 'MM', 'GM', 'MM', 'GM']
```

Внутренние функции

```
abs(-1)
```

```
1
```

```
abs(1)
```

```
1
```

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

```
5
```

```
min([1,2,3,4,5])
```

```
1
```

```
pow(2, 8)
```

```
256
```

```
round(3.37, 1)
```

```
3.4
```

Внутренние функции

```
round(3.37, 1)
```

```
3.4
```

```
sum([1,2,3,4,5])
```

```
15
```

```
h = hex(42)
o = oct(42)
b = bin(42)

print(h)
print(o)
print(b)
```

```
0x2a
0o52
0b101010
```

```
all_true1 = all([True, True, True])
all_true2 = all([True, False, True])

print(all_true1)
print(all_true2)
```

```
True
False
```

```
any_true1 = any([False, False, True])
any_true2 = any([False, False, False])

print(any_true1)
print(any_true2)
```

```
True
False
```

```
players = [('Carlsen', 2842), ('Caruana', 2822),
    ('Mamedyarov', 2801), ('Ding', 2797),
    ('Giri', 2780)]
```

```
any(rating < 2790 for _, rating in players)
```

```
True
```

Внутренние функции

7. Функция zip() объединяет каждый i-ый элемент одного объекта с i-ым элементом остальных до тех пор, пока не закончится самый короткий объект.

```
>>> x = [1, 2, 3]
>>> y = [10, 11]
>>> list(zip(x, y))
[(1, 10), (2, 11)]
```

```
letters = 'abcd'
numbers = (10, 20 , 30)

zipped = zip(letters, numbers)
print(type(zipped))
print(zipped)

zipped_list = list(zipped)
print(zipped_list)
```

```
<class 'zip'>
<zip object at 0x000001FAF30E4308>
[('a', 10), ('b', 20), ('c', 30)]
```

Внутренние функции

```
names = ['Carlsen', 'Caruana', 'Mamedyarov', 'Ding', 'Giri']
ratings = [2842, 2822, 2801, 2797, 2780]

players = dict(zip(names, ratings))
players
```

```
{'Carlsen': 2842,
 'Caruana': 2822,
 'Mamedyarov': 2801,
 'Ding': 2797,
 'Giri': 2780}
```

Функции

```
def greeting():
    '''
    DOCSTRING: Information about the function
    INPUT: no input...
    OUTPUT: Hello!
    '''
    print('Hello!')
```

```
greeting()
```

```
Hello!
```

```
greeting
```

```
<function __main__.greeting()>
```

```
help(greeting)
```

```
Help on function greeting in module __main__:

greeting()
    DOCSTRING: Information about the function
    INPUT: no input...
    OUTPUT: Hello!
```

```
def print_name(name='Default'):
    print(name)
```

```
print_name('Elijah')
```

```
Elijah
```

```
print_name()
```

```
Default
```

```
result = print_name()
print(result)
print(type(result))
```

```
Default
None
<class 'NoneType'>
```

Функции

```
def greeting():
    '''
    DOCSTRING: Information about the function
    INPUT: no input...
    OUTPUT: Hello!
    '''
    print('Hello!')
```

```
greeting()
```

```
Hello!
```

```
greeting
```

```
<function __main__.greeting()>
```

```
help(greeting)
```

```
Help on function greeting in module __main__:

greeting()
    DOCSTRING: Information about the function
    INPUT: no input...
    OUTPUT: Hello!
```

```
def print_name(name='Default'):
    print(name)
```

```
print_name('Elijah')
```

```
Elijah
```

```
print_name()
```

```
Default
```

```
result = print_name()
print(result)
print(type(result))
```

```
Default
None
<class 'NoneType'>
```

Функции

```
def get_greeting(name):
    return 'Hello ' + name

greeting = get_greeting('Elijah')
greeting
```

```
'Hello Elijah'
```

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

result = get_sum(10, 2)
result
```

```
12
```

```
def is_adult(age):
    return age >=18

is_adult = is_adult(20)
is_adult
```

```
True
```

```
def calc_taxes(*args):
    for x in args:
        print(f'Got payment = {x}')
    return sum(args) * 0.06
```

```
calc_taxes(10,20,30)
```

```
Got payment = 10
Got payment = 20
Got payment = 30

3.5999999999999996
```

```
calc_taxes(10,20,30, 40)
```

```
Got payment = 10
Got payment = 20
Got payment = 30
Got payment = 40
```

Функции

```
def save_players(**kwargs):
    for k, v in kwargs.items():
        print(f'Player {k} has rating {v}')

save_players(Carlsen=2800, Giri=2780)
```

```
Player Carlsen has rating 2800
Player Giri has rating 2780
```

Внутренние функции

Как видите, несмотря на то, что x состоит из трех элементов, были использованы только два, так как самый короткий объект — в данном случае y — состоит всего из 2-х элементов.

Приведенное ниже решение работает следующим образом: сперва создается второй список (elements[1:]), который равен исходному списку, но без первого элемента. Затем поочередно сравниваются элементы из этих двух списков и в результате каждого такого сравнения мы получаем True или False. После чего функция all() возвращает результат обработки этого набора True и False.

```
def all_the_same(elements):
    return all(x == y for x, y in zip(elements, elements[1:]))
```

Предположим, наш исходный список elements = [2, 2, 2, 3]. Тогда с помощью zip(), мы объединим полный список ([2, 2, 2, 3]) и список без первого элемента ([2, 2, 3]) таким образом: [(2, 2), (2, 2), (2, 3)], сравнения элементов между собой передадут в функцию all() набор [True, True, False] и в результате мы получим False, что является правильным ответом, так как в исходном списке не все элементы одинаковы.