In [1]:
```
from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"
```

PYTHON 3

Условия, циклы, контейнеры

MIPT 2020

1. Условные операторы (if, elif, else)

In [2]:
```
age = 21; oldage = 12

if age >= 18:
    answer = 'Yes, you can drive'
else:
    answer = "No, you can't drive"

print(answer)
```

```
Yes, you can drive
```

Проверок может быть и несколько сразу:

In [3]:
```
age = 17
country = 'Russia'

if age >= 16 and country == 'USA':
    answer = 'Yes, you can drive'
else:
    if age >= 18 and country == 'Russia':
        answer = 'Yes, you can drive'
    else:
        answer = "No, you can't drive"

print(answer)
```

```
No, you can't drive
```

else + if == elif!

In [4]:
```
age = 17
country = 'Russia'

if age >= 16 and country == 'USA':
    answer = 'Yes, you can drive'
elif age >= 18 and country == 'Russia':
    answer = 'Yes, you can drive'
else:
    answer = "No, you can't drive"

print(answer)
```

```
No, you can't drive
```

2. Тернарный условный оператор

In [5]:
```
#a if condition else b

'true' if True else 'false'
'true' if False else 'false'
a = 0
(10 // a) if a != 0 else 2
```

Out[5]: 'true'

Out[5]: 'false'

Out[5]: 2

In [6]:
```
x = 3
y = 2

z = 3 + x if x < y else y
print(z)

z = 3 + (x if x < y else y)

print(z)
```

```
2
5
```

In [7]:
```
age = 15
'kid' if age < 13 else ('teenager' if age < 18 else 'adult') # unreadable
```

Out[7]: 'teenager'

3. Цикл while

In [8]:
```
greetings = 1

while greetings <= 3:
    print(greetings)
    print('Hello! ' * greetings)
    greetings += 1
```

```
1
Hello!
2
Hello! Hello!
3
Hello! Hello! Hello!
```

In [9]:
```
from math import pi
print(pi)

i = 0

while True:
    if int(pi * (10 ** i)) % 10  == 7:
        break
    i += 1

print(str(i) + "th digit of PI is 7")
```

```
3.141592653589793
13th digit of PI is 7
```

In [10]:
```
S = 0
i = 0

while S < 60:
    digit = int(pi * (10 ** i)) % 10
    if digit == 7:
        continue

    S += digit
    i += 1
print("First "+str(i)+" digits of PI without seven is "+str(S))
```

```
First 13 digits of PI without seven is 61
```

4. Цикл for

In [11]:
```
for i in range(10):
    print(i)
```

```
0
1
2
3
4
5
6
7
8
9
```

In [12]:
```
for i in range(5, 10):
    print(i)
```

```
5
6
7
8
9
```

In [13]:
```
for i in range(5, 14, 3):
    print(i)
```

```
5
8
11
```

In [14]:
```
a = range(3)
for x in a:
    print(x)
for x in a:
    print(x)
```

```
0
1
2
0
1
2
```

Итерирование по коллекциям: как делать **не надо**:

In [15]:
```
names = ['Alice', 'Bob', 'Charley']

for i in range(len(names)):
    print('Hello, ' + names[i])
```

```
Hello, Alice
Hello, Bob
Hello, Charley
```

Итерирование по коллекциям: как делать **надо**:

In [16]:
```
names = ['Alice', 'Bob', 'Charley']
id(names[0]), id(names[1]), id(names[2])

for name in names:
    print('Hello, ' + name, id(name))
```

Out[16]: (139632945654192, 139632945613296, 139632945611056)

```
Hello, Alice 139632945654192
Hello, Bob 139632945613296
Hello, Charley 139632945611056
```

In [17]:
```
names = ['Alice', 'Bob', 'Charley', 'Dave', 'Eve']

for name in names:
    if len(name) < 4:
        continue

    print('Hello, ' + name)
```

```
Hello, Alice
Hello, Charley
Hello, Dave
```

In [18]:
```
names = ['Alice', 'Bob', 'Charley',  'Dave', 'Eve']

for name in names:
    if len(name) < 4:
        break

    print('Hello, ' + name)
```

```
Hello, Alice
```

In [19]:
```
names = ['Alice', 'Bob', 'Charley',  'Dave', 'Eve', 'Op']

for name in names:
    if len(name) < 3:
        break

    print('Hello, ' + name)
else:
    print('Со всеми поздоровались')
```

```
Hello, Alice
Hello, Bob
Hello, Charley
Hello, Dave
Hello, Eve
```

5. Мои первые контейнеры

In [20]:
```
l = ['a', 'b', 3, 6.75, True]
type(l)
print(l)
print(l[3])
l[2] = 4
print(l)
print(*l)
```

Out[20]: list

```
['a', 'b', 3, 6.75, True]
6.75
['a', 'b', 4, 6.75, True]
a b 4 6.75 True
```

In [21]:
```
l = ('a', 'b', 3, 6.75, True)
type(l)
print(l)
print(l[3])
#l[2] = 4
print(*l)
```

Out[21]: tuple

```
('a', 'b', 3, 6.75, True)
6.75
a b 3 6.75 True
```

In [22]:
```
l = []
s = (1,)
l
type(s)
```

Out[22]: []

Out[22]: tuple

In [23]:
```
a = 1, 2, 3
type(a)
a
```

Out[23]: tuple

Out[23]: (1, 2, 3)

Неосторожное обращение с запятой

In [24]:
```
texts = ["text1", "text2", "text3"]
ix = 1,

print(texts[ix])
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-24-4c53a2eca450> in <module>
      2 ix = 1,
      3
----> 4 print(texts[ix])

TypeError: list indices must be integers or slices, not tuple
```

Отрицательные индексы

In [25]:
```
lost = [4, 8, 15, 16, 23, 42]
lost[-1]
lost[-2]
```

Out[25]: 42

Out[25]: 23

Вылезли за пределы

In [26]:
```
lost[10]
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-26-0f7776a0a6ca> in <module>
----> 1 lost[10]

IndexError: list index out of range
```

Неосторожное ображение с ссылками

In [27]:
```
a = [0] * 3
a
for x in a:
    print(id(x))
a[1] = 2
a
for x in a:
    print(id(x))
```

Out[27]: [0, 0, 0]

```
11496608
11496608
11496608
```

Out[27]: [0, 2, 0]

```
11496608
11496672
11496608
```

In [28]:
```
a = [[0] * 3] * 3
a
for x in a:
    print(id(x))
a[0][1] = 2
a
for x in a:
    print(id(x))

b = copy.deepcopy(a)
b[0][1] = 0
b
for x in b:
    print(id(x))
```

Out[28]:
```
[[0, 0, 0], [0, 0, 0], [0, 0, 0]]

139632945901120
139632945901120
139632945901120

[[0, 2, 0], [0, 2, 0], [0, 2, 0]]

139632945901120
139632945901120
139632945901120

---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-28-8c4e4ae68ed8> in <module>
      8     print(id(x))
      9
---> 10 b = copy.deepcopy(a)
     11 b[0][1] = 0
     12 b

NameError: name 'copy' is not defined
```

Срезы (slice, слайсы)

In [29]:
```
tea_party = ('Rabbit', 'Bear', 'Fox', 'Turtle')
type(slice(tea_party[1:2]))

tea_party[1:3] # a включительно, b НЕ включительно
tea_party[:2]
tea_party[3:]
tea_party[:] # Если нет ни a, ни b - выводится список целиком
tea_party[:-1] # Все элементы списка, кроме последнего
tea_party[0:5:2] # а ещё можно задать шаг прохода s - в данном случае элементы с
tea_party[::-1] # а если задать шаг равным -1, список можно развернуть!
tea_party[:1000000000000000]
tea_party[0:2:-1]
```

Out[29]: slice

Out[29]: ('Bear', 'Fox')

Out[29]: ('Rabbit', 'Bear')

Out[29]: ('Turtle',)

Out[29]: ('Rabbit', 'Bear', 'Fox', 'Turtle')

Out[29]: ('Rabbit', 'Bear', 'Fox')

Out[29]: ('Rabbit', 'Fox')

Out[29]: ('Turtle', 'Fox', 'Bear', 'Rabbit')

Out[29]: ('Rabbit', 'Bear', 'Fox', 'Turtle')

Out[29]: ()

In [30]:
```
t_s = slice(0, 2)
tea_party[t_s]
t_s
#t_s2 = slice(tea_party[1:3])
#t_s2
```

Out[30]: ('Rabbit', 'Bear')

Out[30]: slice(0, 2, None)

Изменения по срезам

In [31]:
```
lost = [4, 8, 15, 16, 23, 42]
lost
id(lost)
lost[2:4] = [150, 160, 230]
lost
id(lost)
```

Out[31]: [4, 8, 15, 16, 23, 42]

Out[31]: 139632945273088

Out[31]: [4, 8, 150, 160, 230, 23, 42]

Out[31]: 139632945273088

In [32]:
```
lost[3:4] = [12, 56, 78, 124]
lost
lost[5:8] = [1]
lost
id(lost)
```

Out[32]: [4, 8, 150, 12, 56, 78, 124, 230, 23, 42]

Out[32]: [4, 8, 150, 12, 56, 1, 23, 42]

Out[32]: 139632945273088

Копирование через слайсы

In [33]:
```
lost_copy = lost[:]
lost_copy is lost
```

Out[33]: False

In [34]:
```
import copy

lost_copy = copy.copy(lost)
lost_copy is lost

deep_lost_copy = copy.deepcopy(lost)
deep_lost_copy is lost
```

Out[34]: False

Out[34]: False

In [35]:
```
a = [[[]]]
b = copy.copy(a)
id(a[0]), id(b[0])
id(a[0][0]), id(b[0][0])
b[0][0] += [1]
b
a
```

Out[35]: (139632945270848, 139632945270848)

Out[35]: (139632945273536, 139632945273536)

Out[35]: [[[1]]]

Out[35]: [[[1]]]

In [36]:
```
a = [[[]]]
b = copy.deepcopy(a)
id(a[0]), id(b[0])
id(a[0][0]), id(b[0][0])
b[0][0] += [1]
b
a
```

Out[36]: (139633083625344, 139632945652096)

Out[36]: (139632945663104, 139632945910720)

Out[36]: [[[1]]]

Out[36]: [[[]]]

In [37]:
```
#методы
nums = [1, 3, 5, 7, 7, 7, 5, 3, 1]
print(nums)
nums.append(8)
print(nums)

nums.insert(1, 2)
print(nums)

nums.remove(3)
print(nums)

del nums[4] # nums.pop(4)
print(nums)

sevens = nums.count(7)
print(sevens)

seven_idx = nums.index(7)
print(seven_idx)

nums.sort()
print(nums)

# Забегаем вперед: сортировка по ключу
#nums.sort(key=lambda x: x**2 % 10, reverse=True)
#print(nums)
```

```
[1, 3, 5, 7, 7, 7, 5, 3, 1]
[1, 3, 5, 7, 7, 7, 5, 3, 1, 8]
[1, 2, 3, 5, 7, 7, 7, 5, 3, 1, 8]
[1, 2, 5, 7, 7, 7, 5, 3, 1, 8]
[1, 2, 5, 7, 7, 5, 3, 1, 8]
2
3
[1, 1, 2, 3, 5, 5, 7, 7, 8]
```

In [38]:
```
print(nums)

keys = [x ** 2 % 10 for x in nums]
keys
nums.sort(key=lambda x: x**2 % 10)

nums
nums.sort(key=lambda x: x**2 % 10, reverse=True)
nums
```

```
[1, 1, 2, 3, 5, 5, 7, 7, 8]
```

Out[38]:
```
[1, 1, 4, 9, 5, 5, 9, 9, 4]
```

Out[38]:
```
[1, 1, 2, 8, 5, 5, 3, 7, 7]
```

Out[38]:
```
[3, 7, 7, 5, 5, 2, 8, 1, 1]
```

In [39]:
```
#функции

nums = [1, 3, 5, 7, 7, 7, 5, 3, 1]

len(nums)
sum(nums)
max(nums)
min(nums)

sorted_nums = sorted(nums)
nums
sorted_nums
```

Out[39]: 9

Out[39]: 39

Out[39]: 7

Out[39]: 1

Out[39]: [1, 3, 5, 7, 7, 7, 5, 3, 1]

Out[39]: [1, 1, 3, 3, 5, 5, 7, 7, 7]

Строка - тоже контейнер

In [40]:
```
s = "abacaba"
s[1]
s[2:5]
```

Out[40]: 'b'

Out[40]: 'aca'

In [41]:
```
A = 'Привет, я обычное предложение, во мне есть слова и пробелы'.split()
print(A)

B = ' '.join(A)
print(B)

C = B.split('e')
print(C)

D = 'э'.join(C)
print(D)
B.replace('e', 'э')
```

```
['Привет,', 'я', 'обычное', 'предложение,', 'во', 'мне', 'есть', 'слова', 'и',
'пробелы']
Привет, я обычное предложение, во мне есть слова и пробелы
['Прив', 'т, я обычно', ' пр', 'длож', 'ни', ', во мн', ' ', 'сть слова и проб
', 'лы']
Привэт, я обычноэ прэдложэниэ, во мнэ эсть слова и пробэлы
```

Out[41]: 'Привэт, я обычноэ прэдложэниэ, во мнэ эсть слова и пробэлы'

6. Концепция генераторов

In [42]:
```
x = range(0, 100000000000000000000000000000, 3)
x
type(x)
x[7373747472222222222]
```

Out[42]: range(0, 100000000000000000000000000000, 3)

Out[42]: range

Out[42]: 22121242416666666666

7. Списковые включения (list comprehensions)

In [43]:
```
[x**2 for x in range(10)]
```

Out[43]: [0, 1, 4, 9, 16, 25, 36, 49, 64, 81]

In [44]:
```
[x**2 for x in range(10) if x%2 == 0]
```

Out[44]: [0, 4, 16, 36, 64]

In [45]:
```
# Двойной цикл
sorted([(y, x) for x in range(2, 10) for y in range(x+1, 10) if y%x == 0])
```

Out[45]: [(4, 2), (6, 2), (6, 3), (8, 2), (8, 4), (9, 3)]

In [46]:
```
# Так плохо (почему?)
matrix = [[0]*3]+[[0]*3]+[[0]*3]
id(matrix[0]), id(matrix[1]), id(matrix[2])


# Так хорошо
matrix = [[0]*3 for i in range(3)]
```

Out[46]: (139632944638976, 139632945049152, 139632996657536)

8. Генераторные выражения (generator expressions)

In [47]:
```
a = [x**2 for x in range(10)]
a
type(a)
sum(a)
```

Out[47]: [0, 1, 4, 9, 16, 25, 36, 49, 64, 81]

Out[47]: list

Out[47]: 285

In [48]:
```
a = (x**2 for x in range(10))
a
type(a)
sum(a)
```

Out[48]: <generator object <genexpr> at 0x7efed410f040>

Out[48]: generator

Out[48]: 285

In [49]:
```
# Истощение генератора
a = (x**2 for x in range(10))
a
type(a)
sum(a)
sum(a)
```

Out[49]:
```
<generator object <genexpr> at 0x7efed410f270>
```

Out[49]:
```
generator
```

Out[49]:
```
285
```

Out[49]:
```
0
```

In [50]:
```
a = (x**2 for x in range(5))
a
type(a)
next(a)
next(a)
next(a)
next(a)
next(a)
next(a, -5)
```

Out[50]:
```
<generator object <genexpr> at 0x7efed410f200>
```

Out[50]:
```
generator
```

Out[50]:
```
0
```

Out[50]:
```
1
```

Out[50]:
```
4
```

Out[50]:
```
9
```

Out[50]:
```
16
```

Out[50]:
```
-5
```

In [51]:
```
a = (x**2 for x in range(5))
for x in a:
    print(x)
for x in a:
    print(x)
```

```
0
1
4
9
16
```