Функции

In [17]:

```
from datetime import datetime


def current_seconds():
    """Return current seconds"""
    return datetime.now().second


current_seconds()
```

Out[17]:

```
34
```

In [18]:

```
help(current_seconds)
```

```
Help on function current_seconds in module __main__:

current_seconds()
    Return current seconds
```

SHIFT + TAB

In [ ]:

```
current_seconds()
```

Что такое функция?

In [19]:

```
print(type(current_seconds))
```

```
<class 'function'>
```

In [20]:

```
current_seconds.__name__
```

Out[20]:

```
'current_seconds'
```

In [21]:

```
current_seconds.__doc__
```

Out[21]:

```
'Return current seconds'
```

In [22]:

```
dir(current_seconds)
```

Out[22]:

```
['__annotations__',
 '__call__',
 '__class__',
 '__closure__',
 '__code__',
 '__defaults__',
 '__delattr__',
 '__dict__',
 '__dir__',
 '__doc__',
 '__eq__',
 '__format__',
 '__ge__',
 '__get__',
 '__getattribute__',
 '__globals__',
 '__gt__',
 '__hash__',
 '__init__',
 '__init_subclass__',
 '__kwdefaults__',
 '__le__',
 '__lt__',
 '__module__',
 '__name__',
 '__ne__',
 '__new__',
 '__qualname__',
 '__reduce__',
 '__reduce_ex__',
 '__repr__',
 '__setattr__',
 '__sizeof__',
 '__str__',
 '__subclasshook__']
```

В Python всё — объект 😱

In [23]:

```
a = 5
dir(a)
```

Out[23]:

```
['__abs__',
 '__add__',
 '__and__',
 '__bool__',
 '__ceil__',
 '__class__',
 '__delattr__',
 '__dir__',
 '__divmod__',
 '__doc__',
 '__eq__',
 '__float__',
 '__floor__',
 '__floordiv__',
 '__format__',
 '__ge__',
 '__getattribute__',
 '__getnewargs__',
 '__gt__',
 '__hash__',
 '__index__',
 '__init__',
 '__init_subclass__',
 '__int__',
 '__invert__',
 '__le__',
 '__lshift__',
 '__lt__',
 '__mod__',
 '__mul__',
 '__ne__',
 '__neg__',
 '__new__',
 '__or__',
 '__pos__',
 '__pow__',
 '__radd__',
 '__rand__',
 '__rdivmod__',
 '__reduce__',
 '__reduce_ex__',
 '__repr__',
 '__rfloordiv__',
 '__rlshift__',
 '__rmod__',
 '__rmul__',
 '__ror__',
 '__round__',
 '__rpow__',
 '__rrshift__',
 '__rshift__',
 '__rsub__',
 '__rtruediv__',
 '__rxor__',
 '__setattr__',
 '__sizeof__',
 '__str__',
 '__sub__',
 '__subclasshook__',
 '__truediv__',
 '__trunc__',
 '__xor__',
 'bit_length',
 'conjugate',
 'denominator',
 'from_bytes',
 'imag',
 'numerator',
 'real',
 'to_bytes']
```

In [24]:

```
5 + "hello"
```

```
----------------------------------------------------------------------
-----
TypeError                                 Traceback (most recent call
 last)
<ipython-input-24-ee244f9d0a0e> in <module>
----> 1 5 + "hello"

TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

Хорошо, и что дальше?

In [9]:

```
func = current_seconds
func()
```

Out[9]:

```
11
```

In [10]:

```
func
```

Out[10]:

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

In [11]:

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

def power(a, b):
    return a ** b

def sub(a, b):
    return a - b

key = 'power'

func = {
    'add':   add,
    'power': power,
    'sub':   sub,
}[key]

func(2, 3)
```

Out[11]:

```
8
```

In [12]:

```
current_seconds.secret = "iMh52KgXWwg"
current_seconds.secret
```

Out[12]:

```
'iMh52KgXWwg'
```

In [30]:

```
def func():
    func.counter += 1

func.counter = 0
```

In [31]:

```
for i in range(5):
    func()

func.counter
```

Out[31]:

```
5
```

In [15]:

```
def func():
    if not hasattr(func, 'counter'):
        setattr(func, 'counter', 0)
    func.counter += 1
```

In [16]:

```
for i in range(7):
    func()

func.counter
```

Out[16]:

```
7
```

НИКОГДА не делайте так, как показано ниже.

In [37]:

```
# Внимание на количество аргументов!

def func(a, b):
    pass

func.__code__ = current_seconds.__code__
func()
```

Out[37]:

```
24
```

In [18]:

```
func
```

Out[18]:

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

Как можно и как нельзя вызывать функции?

In [38]:

```
current_seconds(5)
```

```
---------------------------------------------------------------------------
-----
TypeError                                 Traceback (most recent call
 last)
<ipython-input-38-ba2f9d46cb5a> in <module>
----> 1 current_seconds(5)

TypeError: current_seconds() takes 0 positional arguments but 1 was gi
ven
```

In [48]:

```
def func(a, b, c, d):
    print(f"a = {a}; b = {b}; c = {c}; d = {d}")
```

In [49]:

```
func()
```

```
---------------------------------------------------------------------------
-----
TypeError                                 Traceback (most recent call
 last)
<ipython-input-49-bd1982955a12> in <module>
----> 1 func()

TypeError: func() missing 4 required positional arguments: 'a', 'b',
 'c', and 'd'
```

In [50]:

```
func(1, 2, 3, 4)
```

```
a = 1; b = 2; c = 3; d = 4
```

In [51]:

```
func(c=1, b=2, a=3, d=4)
```

```
a = 3; b = 2; c = 1; d = 4
```

In [52]:

```
func(1, 2, d=3, c=4)
```

```
a = 1; b = 2; c = 4; d = 3
```

In [53]:

```
func(1, 3, a=2, d=4)
```

```
---------------------------------------------------------------------------
-----
TypeError                                 Traceback (most recent call
 last)
<ipython-input-53-92a149731d75> in <module>
----> 1 func(1, 3, a=2, d=4)

TypeError: func() got multiple values for argument 'a'
```

In [54]:

```
func(a=1, b=2, 1, 3)
```

```
  File "<ipython-input-54-d13e1488f6b8>", line 1
    func(a=1, b=2, 1, 3)
                  ^
SyntaxError: positional argument follows keyword argument
```

Распаковка аргументов

In [55]:

```
args = (1, 2, 3, 4)
func(*args)
```

```
a = 1; b = 2; c = 3; d = 4
```

In [58]:

```
args = ['str1', 'str2', 'str3']
print(*args)
```

```
str1 str2 str3
```

In [59]:

```
print(args)
```

```
['str1', 'str2', 'str3']
```

In [60]:

```
a = 1
b = 3
c = 4
d = 2

# Сложные логические вычисления аргументов функции...

func(a=a, b=b, c=c, d=d)
```

```
a = 1; b = 3; c = 4; d = 2
```

In [61]:

```
kwargs = {
    'a': 1,
    'b': 3,
    'c': 4,
    'd': 2,
}

func(**kwargs)
```

```
a = 1; b = 3; c = 4; d = 2
```

In [62]:

```
args = (2, 1)
kwargs = {'d': 3, 'c': 4, }

func(*args, **kwargs)
```

```
a = 2; b = 1; c = 4; d = 3
```

In [63]:

```
# Функция, которая принимает все, что угодно

def func(*args, **kwargs):
    pass

func()
func(5, 6, 7)
func([4], 5, b=12, d=6)
func(a=6, b=8)
```

In [64]:

```
def func(a, b, *args, **kwargs):
    print("Function has started.")
    print("a = {}".format(a))
    print("b = {}".format(b))
    print("args = {}".format(args))
    print("kwargs = {}".format(kwargs))
    print("Function has finished.")
```

In [65]:

```
func(1, 4)
```

```
Function has started.
a = 1
b = 4
args = ()
kwargs = {}
Function has finished.
```

In [66]:

```
func(1, 4, 2, 3, f=6, n=7, m=12)
```

```
Function has started.
a = 1
b = 4
args = (2, 3)
kwargs = {'f': 6, 'n': 7, 'm': 12}
Function has finished.
```

Аргументы по ссылке или по значению?

In [1]:

```
%%bash

cat files/example01.cpp
g++ -O2 -o files/example files/example01.cpp
```

```
#include <iostream>
#include <vector>

void vectorAppender1(std::vector<int> a, int b) {
    // passing argument by value
    a.push_back(b);
}

void vectorAppender2(std::vector<int>& a, int b) {
    // passing argument by reference
    a.push_back(b);
}

int main() {
    std::vector<int> v;

    vectorAppender1(v, 1);
    vectorAppender2(v, 2);

    for (const auto& e : v)
        std::cout << e << ' ';
    std::cout << std::endl;

    return 0;
}
```

In [2]:

```
!files/example
```

```
2
```

In [3]:

```
users = [
    ('Michael', '06.08.62'),
    ('Vadim', '23.08.89'),
]

def user_appender(users, u):
    users.append(u)
```

In [4]:

```
u = ('Nastya', '16.01.97')

print("Before:", users)
user_appender(users, u)
print("After: ", users)
```

```
Before: [('Michael', '06.08.62'), ('Vadim', '23.08.89')]
After:  [('Michael', '06.08.62'), ('Vadim', '23.08.89'), ('Nastya', '1
6.01.97')]
```

In [16]:

```
def user_modifier(u_before, u_after):
    u_before = u_after
```

In [18]:

```
user_b = ['Nastya', '16.01.97']
user_a = ['Anton', '04.11.96']

print("Before:", user_b)
user_modifier(user_b, user_a)
print("After: ", user_b)
```

```
Before: ['Nastya', '16.01.97']
After:  ['Nastya', '16.01.97']
```

In [14]:

```
def user_modifier(u_before, u_after):
    u_before[:] = u_after
```

In [15]:

```
user_b = ['Nastya', '16.01.97']
user_a = ['Anton', '04.11.96']

print("Before:", user_b)
user_modifier(user_b, user_a)
print("After: ", user_b)
```

```
Before: ['Nastya', '16.01.97']
After:  ['Anton', '04.11.96']
```

In [31]:

```
#мой пример
def user_modifier(u_before, u_after):
    u_before = u_after
a = 10
b = 50
user_modifier(a, b)
a
```

Out[31]:

```
10
```

Область видимости

Основное правило поиска **LEGB**: Local -> Enclosed -> Global -> Built-in

In [45]:

```
result = "GLOBAL"

def func():
    print("[local]\t\t", result)

func()
```

```
[local]          GLOBAL
```

In [46]:

```
result = "GLOBAL"

def func():
    result = "LOCAL"
    print("[local]\t\t", result)

print("[global]\t", result)
func()
print("[global]\t", result)
```

```
[global]         GLOBAL
[local]          LOCAL
[global]         GLOBAL
```

In [47]:

```
result = "GLOBAL"

def func():
    global result
    result = "LOCAL"
    print("[local]\t\t", result)

print("[global]\t", result)
func()
print("[global]\t", result)
```

```
[global]         GLOBAL
[local]          LOCAL
[global]         LOCAL
```

In [48]:

```python
result = "GLOBAL"

def func():
    print("[local]\t\t", result)
    result = "LOCAL"
    print("[local]\t\t", result)

print("[global]\t", result)
func()
print("[global]\t", result)
```

```
[global]         GLOBAL

---------------------------------------------------------------------------
-----
UnboundLocalError                         Traceback (most recent call
 last)
<ipython-input-48-514e498e5e73> in <module>()
      7 
      8 print("[global]\t", result)
----> 9 func()
     10 print("[global]\t", result)

<ipython-input-48-514e498e5e73> in func()
      2 
      3 def func():
----> 4     print("[local]\t\t", result)
      5     result = "LOCAL"
      6     print("[local]\t\t", result)

UnboundLocalError: local variable 'result' referenced before assignmen
t
```

In [49]:

```python
result = 'GLOBAL'

def func_outer():
    result = 'ENCLOSED'
    print("[enclosed]\t", result)

    def func():
        result = 'LOCAL'
        print("[local]\t\t", result)

    func()
    print("[enclosed]\t", result)

print("[global]\t", result)
func_outer()
print("[global]\t", result)
```

```
[global]         GLOBAL
[enclosed]       ENCLOSED
[local]          LOCAL
[enclosed]       ENCLOSED
[global]         GLOBAL
```

In [50]:

```
result = 'GLOBAL'

def func_outer():
    result = 'ENCLOSED'
    print("[enclosed]\t", result)

    def func():
        global result
        result = 'LOCAL'
        print("[local]\t\t", result)

    func()
    print("[enclosed]\t", result)

print("[global]\t", result)
func_outer()
print("[global]\t", result)
```

```
[global]         GLOBAL
[enclosed]       ENCLOSED
[local]          LOCAL
[enclosed]       ENCLOSED
[global]         LOCAL
```

In [51]:

```
result = 'GLOBAL'

def func_outer():
    result = 'ENCLOSED'
    print("[enclosed]\t", result)

    def func():
        nonlocal result
        result = 'LOCAL'
        print("[local]\t\t", result)

    func()
    print("[enclosed]\t", result)

print("[global]\t", result)
func_outer()
print("[global]\t", result)
```

```
[global]         GLOBAL
[enclosed]       ENCLOSED
[local]          LOCAL
[enclosed]       LOCAL
[global]         GLOBAL
```

Аргументы по-умолчанию

In [52]:

```
def sum_list(a, start_with=0):
    return sum(a[start_with:])

print(sum_list([4, 2, 3]))
print(sum_list([4, 2, 3], start_with=1))
```

```
9
5
```

In [53]:

```
def sum_list(start_with=0, a):
    return sum(a[start_with:])
```

```
  File "<ipython-input-53-acd466c9b88b>", line 1
    def sum_list(start_with=0, a):
                ^
SyntaxError: non-default argument follows default argument
```

Ожидание vs. Реальность

In [32]:

```
def append_one_list(a=[]):
    print("\tBefore:", a)
    a.append(1)
    print("\tAfter: ", a)
```

In [55]:

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

print("Before:", a)
print("=" * 30)
append_one_list(a)
append_one_list(a)
append_one_list(a)
print("=" * 30)
print("After: ", a)
```

```
Before: [1, 2, 3]
==============================
        Before: [1, 2, 3]
        After:  [1, 2, 3, 1]
        Before: [1, 2, 3, 1]
        After:  [1, 2, 3, 1, 1]
        Before: [1, 2, 3, 1, 1]
        After:  [1, 2, 3, 1, 1, 1]
==============================
After:  [1, 2, 3, 1, 1, 1]
```

In [56]:

```
print("Before:", None)
print("=" * 30)
append_one_list()
append_one_list()
append_one_list()
print("=" * 30)
print("After: ", None)
```

```
Before: None
==============================
        Before: []
        After:  [1]
        Before: [1]
        After:  [1, 1]
        Before: [1, 1]
        After:  [1, 1, 1]
==============================
After:  None
```

In [57]:

```
append_one_list.__defaults__
```

Out[57]:

```
([1, 1, 1],)
```

In [58]:

```
def append_one_list(a=None):
    if a is None:
        a = []

    print("\tBefore:", a)
    a.append(1)
    print("\tAfter: ", a)

def append_one_list(a=None):
    a = a or []

    print("\tBefore:", a)
    a.append(1)
    print("\tAfter: ", a)
```

In [59]:

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

print("Before:", a)
print("=" * 30)
append_one_list(a)
append_one_list(a)
append_one_list(a)
print("=" * 30)
print("After: ", a)
```

```
Before: [1, 2, 3]
==============================
        Before: [1, 2, 3]
        After:  [1, 2, 3, 1]
        Before: [1, 2, 3, 1]
        After:  [1, 2, 3, 1, 1]
        Before: [1, 2, 3, 1, 1]
        After:  [1, 2, 3, 1, 1, 1]
==============================
After:  [1, 2, 3, 1, 1, 1]
```

In [60]:

```
print("Before:", None)
print("=" * 30)
append_one_list()
append_one_list()
append_one_list()
print("=" * 30)
print("After: ", None)
```

```
Before: None
==============================
        Before: []
        After:  [1]
        Before: []
        After:  [1]
        Before: []
        After:  [1]
==============================
After:  None
```

Элементы функционального программирования

Анонимные функции (или lambda-функции)

In [61]:

```
result = {
    'a': 1,
    'b': 3,
    'c': 2,
    'd': 5,
    'f': 4,
}
```

In [62]:

```python
sorted(result.items(), reverse=True)
```

Out[62]:

```
[('f', 4), ('d', 5), ('c', 2), ('b', 3), ('a', 1)]
```

In [63]:

```python
def func_key(pair):
    return pair[1]

print(type(func_key))

sorted(result.items(), key=func_key, reverse=True)
```

```
<class 'function'>
```

Out[63]:

```
[('d', 5), ('f', 4), ('b', 3), ('c', 2), ('a', 1)]
```

In [64]:

```python
sorted(result.items(), key=lambda pair: pair[1], reverse=True)
```

Out[64]:

```
[('d', 5), ('f', 4), ('b', 3), ('c', 2), ('a', 1)]
```

Мнемоническое правило:

```python
def <lambda>(pair):
    return pair[1]
```

In [65]:

```python
func = lambda pair: pair[1]
print(type(func))
```

```
<class 'function'>
```

In [66]:

```python
from operator import itemgetter

sorted(result.items(), key=itemgetter(1), reverse=True)
```

Out[66]:

```
[('d', 5), ('f', 4), ('b', 3), ('c', 2), ('a', 1)]
```

In [67]:

```python
a = list(range(-5, 5))
a
```

Out[67]:

```
[-5, -4, -3, -2, -1, 0, 1, 2, 3, 4]
```

In [68]:

```
sorted(a, key=lambda x: x**2)
```

Out[68]:

```
[0, -1, 1, -2, 2, -3, 3, -4, 4, -5]
```

In [69]:

```
import functools

func = functools.partial(sorted, key=itemgetter(1))
func(result.items())
```

Out[69]:

```
[('a', 1), ('c', 2), ('b', 3), ('f', 4), ('d', 5)]
```

In [70]:

```
[lambda x: x ** 2,
 lambda x, y: x < y,
 lambda s: s.strip().split()]
```

Out[70]:

```
[<function __main__.<lambda>(x)>,
 <function __main__.<lambda>(x, y)>,
 <function __main__.<lambda>(s)>]
```

Функция map

In [71]:

```
result_a = range(10)
result_b = map(lambda x: x ** 2, result_a)

print(list(result_a))
print(list(result_b))

result_b
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

Out[71]:

```
<map at 0x7f254003bc18>
```

In [72]:

```
result = [
    ('a', 1),
    ('b', 3),
    ('c', 2),
    ('d', 5),
    ('f', 4),
]

list(map(itemgetter(0), result))
```

Out[72]:

```
['a', 'b', 'c', 'd', 'f']
```

In [73]:

```
result = '1,2,3,4,5,6\n'

list(map(int, result.split(',')))
```

Out[73]:

```
[1, 2, 3, 4, 5, 6]
```

In [74]:

```
list(map(ord, 'education'))
```

Out[74]:

```
[101, 100, 117, 99, 97, 116, 105, 111, 110]
```

In [75]:

```
message = """1 2
2 3
3 4
4 5"""
print(message, file=open("/tmp/filename", "w"))
```

In [76]:

```
with open("/tmp/filename", "r") as f_name:
    for i, j in map(lambda s: s.strip().split(), f_name):
        i, j = map(int, [i, j])
        print(i, j)
```

```
1 2
2 3
3 4
4 5
```

In [77]:

```
with open("/tmp/filename", "r") as f_name:
    for i, j in map(str.split, map(str.strip, f_name)):
        i, j = map(int, [i, j])
        print(i, j)
```

```
1 2
2 3
3 4
4 5
```

In [78]:

```
import operator
```

In [79]:

```
result_a = [5, 6, 7]
result_b = [4, 5, 6]

list(map(operator.add, result_a, result_b))
```

Out[79]:

```
[9, 11, 13]
```

In [80]:

```
result = [(5, 4), (6, 5), (7, 6)]
```

In [81]:

```
list(map(lambda x, y: x + y, result))
```

```
---------------------------------------------------------------------------
-----
TypeError                                 Traceback (most recent call
 last)
<ipython-input-81-cd0bf2c166d0> in <module>()
----> 1 list(map(lambda x, y: x + y, result))

TypeError: <lambda>() missing 1 required positional argument: 'y'
```

In [82]:

```
list(map(lambda (x, y): x + y, result))
```

```
  File "<ipython-input-82-988a486c5318>", line 1
    list(map(lambda (x, y): x + y, result))
                    ^
SyntaxError: invalid syntax
```

В стандарте больше нет поддержки кортежей как аргументов, см. [PEP-3113 (https://www.python.org/dev/peps/pep-3113/)](https://www.python.org/dev/peps/pep-3113/).

In [83]:

```
%%python2

from __future__ import print_function

result = [(5, 4), (6, 5), (7, 6)]
result = map(lambda (x, y): x + y, result)
print(*result)
```

```
9 11 13
```

In [84]:

```
list(map(lambda p: p[0] + p[1], result))
```

Out[84]:

```
[9, 11, 13]
```

In [85]:

```
from itertools import starmap

list(starmap(operator.add, result))
```

Out[85]:

```
[9, 11, 13]
```

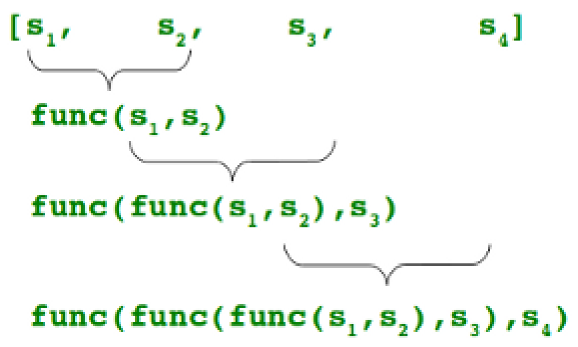
Функция reduce

In [86]:

```
from functools import reduce
```

In [87]:

```
def my_reduce(func, seq):
    res = seq[0]
    for elem in seq[1:]:
        res = func(res, elem)
    return res
```



In [88]:

```
print(my_reduce(lambda x, y: x + y, [1, 2, 3, 4]))
print(my_reduce(lambda x, y: x * y, [1, 2, 3, 4]))
```

```
10
24
```

In [89]:

```
print(reduce(lambda x, y: x + y, [1, 2, 3, 4]))
print(reduce(lambda x, y: x * y, [1, 2, 3, 4]))
```

```
10
24
```

In [90]:

```
from operator import add, mul

print(reduce(add, [1, 2, 3, 4]))
print(reduce(mul, [1, 2, 3, 4]))
```

```
10
24
```

Функция filter

In [91]:

```
filter(lambda x: x > 0, range(-5, 5))
```

Out[91]:

```
<filter at 0x7f2540709588>
```

In [92]:

```
list(filter(lambda x: x > 0, range(-5, 5)))
```

Out[92]:

```
[1, 2, 3, 4]
```

In [93]:

```
list(filter(lambda x: x % 2, range(-5, 5)))
```

Out[93]:

```
[-5, -3, -1, 1, 3]
```

In [94]:

```
list(filter(lambda x: x not in {'п', 'л'}, "параллелепипед"))
```

Out[94]:

```
['а', 'р', 'а', 'е', 'е', 'и', 'е', 'д']
```

In [95]:

```
result = {
    'key1': 1,
    'key2': 2,
    'key3': 3,
    'art': 'Hermitage',
    'ord': 7,
}
```

In [96]:

```
{k: result[k] for k in filter(lambda k: k.startswith('key'), result)}
```

Out[96]:

```
{'key1': 1, 'key2': 2, 'key3': 3}
```

In [97]:

```
dict(filter(lambda p: p[0].startswith('key'), result.items()))
```

Out[97]:

```
{'key1': 1, 'key2': 2, 'key3': 3}
```

In [98]:

```
{k: v for k, v in result.items() if k.startswith('key')}
```

Out[98]:

```
{'key1': 1, 'key2': 2, 'key3': 3}
```

Функция zip

In [99]:

```
zip(range(10), "параллелепипед")
```

Out[99]:

```
<zip at 0x7f254002c1c8>
```

In [100]:

```
list(zip(range(10), "параллелепипед"))
```

Out[100]:

```
[(0, 'п'),
 (1, 'а'),
 (2, 'р'),
 (3, 'а'),
 (4, 'л'),
 (5, 'л'),
 (6, 'е'),
 (7, 'л'),
 (8, 'е'),
 (9, 'п')]
```

In [101]:

```
list(zip(
    "параллелепипед",
    range(10),
    [True, True, True, False, False, True, False]
))
```

Out[101]:

```
[('п', 0, True),
 ('а', 1, True),
 ('р', 2, True),
 ('а', 3, False),
 ('л', 4, False),
 ('л', 5, True),
 ('е', 6, False)]
```

In [102]:

```
s = "параллелепипед"
list(zip(range(len(s)), s))
```

Out[102]:

```
[(0, 'п'),
 (1, 'а'),
 (2, 'р'),
 (3, 'а'),
 (4, 'л'),
 (5, 'л'),
 (6, 'е'),
 (7, 'л'),
 (8, 'е'),
 (9, 'п'),
 (10, 'и'),
 (11, 'п'),
 (12, 'е'),
 (13, 'д')]
```

```
In [103]:
```

```python
enumerate("параллелепипед")
```

```
Out[103]:

<enumerate at 0x7f2540706048>
```

```
In [104]:
```

```python
list(enumerate("параллелепипед"))
```

```
Out[104]:

[(0, 'п'),
 (1, 'а'),
 (2, 'р'),
 (3, 'а'),
 (4, 'л'),
 (5, 'л'),
 (6, 'е'),
 (7, 'л'),
 (8, 'е'),
 (9, 'п'),
 (10, 'и'),
 (11, 'п'),
 (12, 'е'),
 (13, 'д')]
```

```
In [105]:
```

```python
for i, c in enumerate("параллелепипед"):
    print(i, '\t', c)
```

```
0        п
1        а
2        р
3        а
4        л
5        л
6        е
7        л
8        е
9        п
10       и
11       п
12       е
13       д
```

In [106]:

```
from itertools import zip_longest

list(zip_longest(range(10), "параллелепипед"))
```

Out[106]:

```
[(0, 'п'),
 (1, 'а'),
 (2, 'р'),
 (3, 'а'),
 (4, 'л'),
 (5, 'л'),
 (6, 'е'),
 (7, 'л'),
 (8, 'е'),
 (9, 'п'),
 (None, 'и'),
 (None, 'п'),
 (None, 'е'),
 (None, 'д')]
```

In [107]:

```
list(zip_longest(range(10), "параллелепипед", fillvalue=-1))
```

Out[107]:

```
[(0, 'п'),
 (1, 'а'),
 (2, 'р'),
 (3, 'а'),
 (4, 'л'),
 (5, 'л'),
 (6, 'е'),
 (7, 'л'),
 (8, 'е'),
 (9, 'п'),
 (-1, 'и'),
 (-1, 'п'),
 (-1, 'е'),
 (-1, 'д')]
```

In [108]:

```
list(zip_longest(
    "параллелепипед",
    range(10),
    [True, True, True, False, False, True, False]
))
```

Out[108]:

```
[('п', 0, True),
 ('а', 1, True),
 ('р', 2, True),
 ('а', 3, False),
 ('л', 4, False),
 ('л', 5, True),
 ('е', 6, False),
 ('л', 7, None),
 ('е', 8, None),
 ('п', 9, None),
 ('и', None, None),
 ('п', None, None),
 ('е', None, None),
 ('д', None, None)]
```

In [109]:

```
result = list(zip(
    "параллелепипед",
    range(-5, 5),
    [True, True, True, False, False, True, False]
))

result
```

Out[109]:

```
[('п', -5, True),
 ('а', -4, True),
 ('р', -3, True),
 ('а', -2, False),
 ('л', -1, False),
 ('л', 0, True),
 ('е', 1, False)]
```

In [110]:

```
list(map(itemgetter(1), result))
```

Out[110]:

```
[-5, -4, -3, -2, -1, 0, 1]
```

In [111]:

```
[list(map(itemgetter(i), result)) for i in range(len(result[0]))]
```

Out[111]:

```
[['п', 'а', 'р', 'а', 'л', 'л', 'е'],
 [-5, -4, -3, -2, -1, 0, 1],
 [True, True, True, False, False, True, False]]
```

In [112]:

```
list(zip(*result))
```

Out[112]:

```
[('п', 'а', 'р', 'а', 'л', 'л', 'е'),
 (-5, -4, -3, -2, -1, 0, 1),
 (True, True, True, False, False, True, False)]
```

Разбор домашки! 😲

Задача B. Сложная сортировка

Дан массив $a_1, \ldots, a_n$ из n натуральных чисел.

Требуется отсортировать числа в массиве в порядке возрастания суммы цифр, а при равной сумме цифр — по возрастанию самого числа.

Олимпиада "Я — профессионал". Направление "Программирование и ИТ". Онлайн этап. 2019 г.

In [113]:

```
s = "14 23 22"

sorted(s.split(), key=lambda n: (sum(map(int, n)), int(n)))
```

Out[113]:

```
['22', '14', '23']
```

Задача D. За линией фронта

Роберт успешно справился с вылазкой в тыл врага и раздобыл одну полоску из тетради в клетку длиной N клеток. Кроме того в руки Роберта попал лист изменений этой полоски. Враг выбирал две позиции i и j и записывал между ними новое сообщение (оба конца/позиции включены). При этом, если новое сообщение накладывалось на старое, то старое становилось навсегда утерянным.

Напишите программу, которая определит, сколько на полоске осталось не утерянных сообщений.

Онлайн вступительные в Финтех Школу Тинькофф. Весна. 2019 г.

In [114]:

```
s = """
8 10
2 9
1 3
"""
s = s.strip().split('\n')
```

In [115]:

```
messages = []
for line in s:
    i, j = map(int, line.split())
    messages = filter(lambda m: not (m[0] <= i <= m[1] or
                                     m[0] <= j <= m[1] or
                                     i <= m[0] and m[1] <= j), messages)
    messages = list(messages)
    messages.append((i, j))
print(len(messages))
```

```
1
```

In [116]:

```
from itertools import filterfalse

messages = []
for line in s:
    i, j = map(int, line.split())
    messages = filterfalse(lambda m: m[0] <= i <= m[1] or
                                     m[0] <= j <= m[1] or
                                     i <= m[0] and m[1] <= j, messages)
    messages = list(messages)
    messages.append((i, j))
print(len(messages))
```

```
1
```

Декораторы

In [117]:

```
def decorator(func):
    return func

@decorator
def greetings():
    return "Hello world!"

print(greetings())

greetings.__name__
```

```
Hello world!
```

Out[117]:

```
'greetings'
```

In [118]:

```
def decorator(func):
    def func_new():
        return "Bonjour le monde!"
    return func_new

@decorator
def greetings():
    return "Hello world!"

print(greetings())

greetings.__name__
```

```
Bonjour le monde!
```

Out[118]:

```
'func_new'
```

In [119]:

```
def logger(func):
    def wrapper(a):
        result = func(a)
        with open('/tmp/decorator.logs', 'a') as f_output:
            # Способ 1 писать в файл
            # f_output.write("num = {}; result = {}\n".format(len(a), result))

            # Способ 2 писать в файл
            print("num = {}; result = {}".format(len(a), result), file=f_output)
        return result
    return wrapper

@logger
def summator(a):
    return sum(a)
```

In [120]:

```
summator([1, 5, 3, 0])
```

Out[120]:

```
9
```

In [121]:

```
!cat /tmp/decorator.logs
```

```
num = 4; result = 9
```

In [122]:

```
def summator(a):
    return sum(a)

logger(summator)([5, 5, 5])
```

Out[122]:

```
15
```

In [123]:

```
def logger(func):
    def wrapper(*args, **argv):
        result = func(*args, **argv)
        with open('/tmp/decorator.logs', 'a') as f_output:
            f_output.write("func = \"{}\"; result = {}\n".format(func.__name__, res
        return result
    return wrapper

@logger
def summator(a):
    return sum(a)

@logger
def mod_taker(a, mod):
    return list(map(lambda x: x % mod, a))
```

In [124]:

```
summator([1, 2, 3, 4])
```

Out[124]:

```
10
```

In [125]:

```
mod_taker([1, 2, 3, 4], 3)
```

Out[125]:

```
[1, 2, 0, 1]
```

In [126]:

```
!cat /tmp/decorator.logs
```

```
num = 4; result = 9
num = 3; result = 15
func = "summator"; result = 10
func = "mod_taker"; result = [1, 2, 0, 1]
```

In [127]:

```
summator.__name__
```

Out[127]:

```
'wrapper'
```

In [128]:

```
import functools

def logger(func):
    @functools.wraps(func)
    def wrapper(*args, **argv):
        result = func(*args, **argv)
        with open('/tmp/decorator.logs', 'a') as f_output:
            f_output.write("func = \"{}\"; result = {}\n".format(func.__name__, res
        return result
    return wrapper

@logger
def summator(a):
    return sum(a)
```

In [129]:

```
summator.__name__
```

Out[129]:

```
'summator'
```

In [130]:

```
def logger(filename):
    def decorator(func):
        def wrapper(*args, **argv):
            result = func(*args, **argv)
            with open(filename, 'a') as f_output:
                f_output.write("func = \"{}\"; result = {}\n".format(func.__name__,
            return result
        return wrapper
    return decorator

@logger("/tmp/decorator2.logs")
def summator(a):
    return sum(a)

summator([1, 2, 3, 5])
```

Out[130]:

```
11
```

In [131]:

```
!cat /tmp/decorator2.logs
```

```
func = "summator"; result = 11
```

In [132]:

```python
from time import sleep

def cached(func):
    cache = dict()
    @functools.wraps(func)
    def wrapper(*args):
        key = (func, args)
        if key not in cache:
            cache[key] = func(*args)
        return cache[key]
    return wrapper

@cached
def power2(x):
    sleep(3)
    return 2 ** x

print(power2(8))
print(power2(8))
print(power2(4))
print(power2(8))
print(power2(4))
```

```
256
256
16
256
16
```

In [133]:

```python
from functools import lru_cache

@lru_cache(maxsize=5)
def power2(x):
    sleep(3)
    return 2 ** x

print(power2(8))
print(power2(8))
print(power2(4))
print(power2(8))
print(power2(4))
```

```
256
256
16
256
16
```

In [134]:

```
def decorator1(func):
    def wrapped():
        print('Entering 1st decorator...')
        result = func()
        print('Exiting 1st decorator...')
        return result
    return wrapped

def decorator2(func):
    def wrapped():
        print('Entering 2nd decorator...')
        result = func()
        print('Exiting 2nd decorator...')
        return result
    return wrapped

@decorator1
@decorator2
def greetings():
    print("Hello world!")

greetings()
```

```
Entering 1st decorator...
Entering 2nd decorator...
Hello world!
Exiting 2nd decorator...
Exiting 1st decorator...
```

In [ ]:

```
```