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

PYTHON

Seminar 5

libraries

https://docs.python.org/3/py-modindex.html (https://docs.python.org/3/py-modindex.html) - все модули в питоне, их очень много

argparse

In [2]:
```
code = """

import argparse

parser = argparse.ArgumentParser(description='Process some integers.')
parser.add_argument('integers', metavar='N', type=int, nargs='+',
                    help='an integer for the accumulator')
parser.add_argument('--sum', dest='accumulate', action='store_const',
                    const=sum, default=max,
                    help='sum the integers (default: find the max)')

args = parser.parse_args()
print(args.accumulate(args.integers))

"""

with open("summator.py", "w") as out:
    out.write(code)
```

Out[2]:
```
465
```

In [3]:
```
import os
import stat

st = os.stat("summator.py")
os.chmod("summator.py", st.st_mode | stat.S_IEXEC)
```

In [4]:
```
!python3 summator.py --sum 1 2 3 4
```

```
10
```

In [5]:
```
!python3 summator.py 1 2 3 4
```

```
4
```

In [6]:
```
!python3 summator.py --help
```

```
usage: summator.py [-h] [--sum] N [N ...]

Process some integers.

positional arguments:
  N           an integer for the accumulator

optional arguments:
  -h, --help  show this help message and exit
  --sum       sum the integers (default: find the max)
```

In [7]:
```
import argparse

parser = argparse.ArgumentParser(description='some description')
parser.add_argument(
    dest='parsed_name', # argument can be obtained by parser.parser_name
    action='store',  # action with argument (default)
    metavar='Name',  # in help this argument will be seen as Name
    type=int,        # type of given argument
    nargs='?',       # multiple arguments will be stored for +. Also can be '?'
    const='some name', # if 'name' was specified, but no value provided
    default='default', # if 'name' was not specified
    help='name of a value', # will be printed in help
)

args = parser.parse_args('1'.split())
args
```

Out[7]:
```
_StoreAction(option_strings=[], dest='parsed_name', nargs='?', const='some name
', default='default', type=<class 'int'>, choices=None, help='name of a value',
metavar='Name')
```

Out[7]:
```
Namespace(parsed_name=1)
```

array

In [8]:
```
from array import array
import sys

a = array('i', [1, 2, 3])
print(f'int size {sys.getsizeof(1)}')
a.itemsize      # not 28 huh?
a.buffer_info() # address, length
```

```
int size 28
```

Out[8]:
```
4
```

Out[8]:
```
(140646235706016, 3)
```

Все те же операции, что и с листом, и еще есть другие

In [9]:
```
b = array('u', 'lalala ')
b.fromunicode('hello world')
b
b.tounicode()
```

Out[9]:
```
array('u', 'lalala hello world')
```

Out[9]:
```
'lalala hello world'
```

```
In [10]: b.itemsize
```

```
Out[10]: 4
```

отступление

```
In [11]: def привет():
             print('Привет!')

         привет()
```

```
Привет!
```

bisect

```
In [12]: from bisect import bisect_right, bisect_left

         a = [1, 1, 1, 2, 3, 4, 5, 8, 9, 10]
         bisect_right(a, 1)     # Индекс первого значения, большего данного
         bisect_left(a, 7) # Индекс  первого значения, больше или равного данному
```

```
Out[12]: 3

Out[12]: 7
```

copy

```
In [13]: import copy

         a = [1, 2, [3, 4]]
         b = copy.copy(a)
         c = copy.deepcopy(a)

         c[2][0] = 10
         c, a

         b[2][0] = 10
         b, a
```

```
Out[13]: ([1, 2, [10, 4]], [1, 2, [3, 4]])

Out[13]: ([1, 2, [10, 4]], [1, 2, [10, 4]])
```

datetime

```
In [14]: import datetime

         a = datetime.datetime.now()
         ...
         b = datetime.datetime.now() - a
         a
         b
         a.year, a.month, a.day, a.weekday()
         b.total_seconds()

Out[14]: Ellipsis

Out[14]: datetime.datetime(2020, 4, 15, 7, 13, 55, 768010)

Out[14]: datetime.timedelta(microseconds=3459)

Out[14]: (2020, 4, 15, 2)

Out[14]: 0.003459
```

```
In [15]: c = a - datetime.timedelta(366, 0, 0) # минус год это не всегда минус 365 дней
         c

Out[15]: datetime.datetime(2019, 4, 15, 7, 13, 55, 768010)
```

Больше - **dateutil**

dis

```
In [16]: import dis

         def f():
             return 0

         dis.dis(f)
         instrs = dis.get_instructions(f)
         for instr in instrs:
             instr

           4           0 LOAD_CONST               1 (0)
                       2 RETURN_VALUE

Out[16]: Instruction(opname='LOAD_CONST', opcode=100, arg=1, argval=0, argrepr='0', offs
         et=0, starts_line=4, is_jump_target=False)

Out[16]: Instruction(opname='RETURN_VALUE', opcode=83, arg=None, argval=None, argrepr=
         '', offset=2, starts_line=None, is_jump_target=False)
```

```
In [17]: def g():
             a = 5
             a += 2
             b = a
             return b

         dis.dis(g)
```

```
  2           0 LOAD_CONST               1 (5)
              2 STORE_FAST               0 (a)

  3           4 LOAD_FAST                0 (a)
              6 LOAD_CONST               2 (2)
              8 INPLACE_ADD
             10 STORE_FAST               0 (a)

  4          12 LOAD_FAST                0 (a)
             14 STORE_FAST               1 (b)

  5          16 LOAD_FAST                1 (b)
             18 RETURN_VALUE
```

enum

```
In [18]: from enum import Enum

         class Color(Enum):
             GREEN = 1
             YELLOW = 2
             RED = 3

         Color(2)
         Color.RED.value
         Color['RED']
         Color.RED + Color.GREEN
```

```
Out[18]: <Color.YELLOW: 2>

Out[18]: 3

Out[18]: <Color.RED: 3>

         ---------------------------------------------------------------------------
         TypeError                                 Traceback (most recent call last)
         <ipython-input-18-2c0ce0af9981> in <module>
               9 Color.RED.value
              10 Color['RED']
         ---> 11 Color.RED + Color.GREEN

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

```
In [19]: d = {
             Color.RED: 1,
             Color.YELLOW: 5
         }
         d[Color.RED]
         hash(Color.RED)
```

```
Out[19]: 1

Out[19]: -3874750920409317548
```

```
In [20]: Color.RED.value = 5
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-20-1c3cf13ed98b> in <module>
----> 1 Color.RED.value = 5

/usr/lib/python3.8/types.py in __set__(self, instance, value)
    180     def __set__(self, instance, value):
    181         if self.fset is None:
--> 182             raise AttributeError("can't set attribute")
    183         self.fset(instance, value)
    184 

AttributeError: can't set attribute
```

fractions

```
In [21]: from fractions import Fraction
         Fraction(2, 3) + Fraction(3, 2)
         Fraction(7, 9) * 99999999999
         Fraction(2, 3) ** Fraction(15, 7)
```

```
Out[21]: Fraction(13, 6)

Out[21]: Fraction(77777777777, 1)

Out[21]: 0.41943202552688796
```

```
In [22]: Fraction(3.1415926535)
```

```
Out[22]: Fraction(1768559437956561, 562949953421312)
```

```
In [23]: 1768559437956561 / 562949953421312
```

```
Out[23]: 3.1415926535
```

```
In [24]: Fraction(3.1415926535).limit_denominator(50)
```

```
Out[24]: Fraction(22, 7)
```

```
In [25]: (-2) ** (1 / 3) == (-2) ** (2 / 6)
```

```
Out[25]: True
```

functools

```
In [26]: import functools
         import time

         @functools.lru_cache(maxsize=3)
         def heavy(a):
             time.sleep(1)
             return a

         s = datetime.datetime.now()
         heavy(1)
         (datetime.datetime.now() - s).total_seconds()

         s = datetime.datetime.now()
         heavy(1)
         (datetime.datetime.now() - s).total_seconds()

Out[26]: 1

Out[26]: 1.005649

Out[26]: 1

Out[26]: 0.004754
```

```
In [27]: heavy.cache_info()

Out[27]: CacheInfo(hits=1, misses=1, maxsize=3, currsize=1)
```

```
In [28]: # generic functions

         @functools.singledispatch
         def f(arg):
             print('default')

         @f.register(int)
         def _(arg):
             print('int')

         f(1)
         f('2')

         int
         default
```

```
In [29]: def wrapper(f):
             @functools.wraps(f)
             def internal(*args, **kwargs):
                 print('start function')
                 f(*args, **kwargs)
                 print('end function')
             return internal

         @wrapper
         def adder(a, b):
             print(a + b)

         adder(1, 2)
         adder.__name__

         start function
         3
         end function

Out[29]: 'adder'
```

gc

```
In [30]: import gc


         sys.getallocatedblocks()
         gc.collect()
         sys.getallocatedblocks()

Out[30]: 252550

Out[30]: 1176

Out[30]: 250503
```

```
In [31]: sys.getallocatedblocks()
         a = list(range(10000))
         sys.getallocatedblocks()
         del a
         sys.getallocatedblocks()
         gc.collect()
         sys.getallocatedblocks()

Out[31]: 251031

Out[31]: 260402

Out[31]: 250834

Out[31]: 514

Out[31]: 250621
```

```
In [32]: gc.disable()
         gc.isenabled()

         gc.enable()
         gc.isenabled()

Out[32]: False

Out[32]: True
```

hashlib

```
In [33]: import hashlib

         hashlib.algorithms_guaranteed

Out[33]: {'blake2b',
          'blake2s',
          'md5',
          'sha1',
          'sha224',
          'sha256',
          'sha384',
          'sha3_224',
          'sha3_256',
          'sha3_384',
          'sha3_512',
          'sha512',
          'shake_128',
          'shake_256'}
```

```
In [34]: algo = hashlib.sha256()
         algo.update(b'hehehe nobody knows this')
         algo.hexdigest()

Out[34]: '7989bcd02e6640c670d78456850099c0e57c933c65121bd9d1fc9b4b26504919'
```

itertools

```
In [35]: import itertools


         for x in itertools.repeat(12, 3):
             print(x, end=' ')

12 12 12
```

```
In [36]: for i, x in enumerate(itertools.cycle('abcd')):
             if i == 10:
                 break
             print(x, end=' ')

a b c d a b c d a b
```

```
In [37]: for i, x in enumerate(itertools.count(5)):
             if i == 10:
                 break
             print(x, end=' ')

5 6 7 8 9 10 11 12 13 14
```

```
In [38]: a = [1, 2, 3]
         b = [4, 5, 6]
         for x in itertools.chain(a, b):
             print(x, end=' ')

1 2 3 4 5 6
```

```
In [39]: for x in itertools.starmap(sum, [[(1, 2)], [(3, 4, 5)]]): # sum((1, 2))
             print(x, end=' ')

3 12
```

```
In [40]: for perm in itertools.permutations('abc', 2):
             print(perm, end=' ')

('a', 'b') ('a', 'c') ('b', 'a') ('b', 'c') ('c', 'a') ('c', 'b')
```

json

In [41]:
```
import json

my_json = '''
{
    "a": 5,
    "b": "ololo"
}
'''

my_json
json.loads(my_json)
json.dumps(json.loads(my_json))
```

Out[41]:
```
'\n{\n    "a": 5,\n    "b": "ololo"\n}\n'
```

Out[41]:
```
{'a': 5, 'b': 'ololo'}
```

Out[41]:
```
'{"a": 5, "b": "ololo"}'
```

logging

In [42]:
```
import logging

logger = logging.getLogger(__name__)

logger.info('everything ok')
logger.warning('well yes but actually no')
logger.error('something went wrong')
try:
    raise ValueError
except Exception:
    logger.exception('total crush')
```

```
well yes but actually no
something went wrong
total crush
Traceback (most recent call last):
  File "<ipython-input-42-0986fb87e585>", line 9, in <module>
    raise ValueError
ValueError
```

In [43]:
```
__name__
```

Out[43]:
```
'__main__'
```

math

In [44]:
```
import math

math.sqrt(10)
math.pow(2, 3.5)
```

Out[44]:
```
3.1622776601683795
```

Out[44]:
```
11.313708498984761
```

In [45]:
```
math.gcd(100, 250)
```

Out[45]:
```
50
```

```
In [46]: math.pi
         math.e

Out[46]: 3.141592653589793

Out[46]: 2.718281828459045
```

```
In [47]: math.factorial(5)

Out[47]: 120
```

```
In [48]: sum([.1, .1, .1, .1, .1, .1, .1, .1, .1, .1])
         math.fsum([.1, .1, .1, .1, .1, .1, .1, .1, .1, .1])

Out[48]: 0.9999999999999999

Out[48]: 1.0
```

```
In [49]: math.isnan(float('nan'))
         math.isinf(float('-inf'))
         math.inf

Out[49]: True

Out[49]: True

Out[49]: inf
```

```
In [50]: math.gamma(6) # Gamma(n) = (n - 1)!

Out[50]: 120.0
```

os

```
In [51]: import os


         with open(os.devnull, 'w') as devnull:
             a = 'aa' * 10000
             for i in range(1000000):
                 _ = devnull.write(a)
         os.devnull

Out[51]: '/dev/null'
```

```
In [52]: os.name
         os.getcwd()
         os.getlogin()
         os.getpid()

Out[52]: 'posix'

Out[52]: '/home/pavel/study/PythonSeminars/seminars/05.5_libraries'

Out[52]: 'pavel'

Out[52]: 5668
```

pathlib

```
In [53]: import pathlib

         cur = pathlib.Path('.')
         test = cur / 'test_dir'
         test.absolute()
         test.stat()

         list(test.iterdir())

         test.is_dir()
```

```
Out[53]: PosixPath('/home/pavel/study/PythonSeminars/seminars/05.5_libraries/test_dir')

         ---------------------------------------------------------------------------
         FileNotFoundError                         Traceback (most recent call last)
         <ipython-input-53-1aa26f6f36b1> in <module>
               4 test = cur / 'test_dir'
               5 test.absolute()
         ----> 6 test.stat()
               7
               8 list(test.iterdir())

         /usr/lib/python3.8/pathlib.py in stat(self)
            1174         os.stat() does.
            1175         """
         -> 1176         return self._accessor.stat(self)
            1177
            1178     def owner(self):

         FileNotFoundError: [Errno 2] No such file or directory: 'test_dir'
```

pickle

```
In [54]: import pickle

         s = {
             'hmm': 'object???',
         }
         with open('really.txt', 'wb') as out:
             pickle.dump(s, out)

         with open('really.txt', 'rb') as inp:
             pickle.load(inp)
             _ = inp.seek(0, 0)
             inp.read()

         pickle.dumps(s)
```

```
Out[54]: {'hmm': 'object???'}

Out[54]: b'\x80\x04\x95\x16\x00\x00\x00\x00\x00\x00\x00}\x94\x8c\x03hmm\x94\x8c\tobjec
         t???\x94s.'

Out[54]: b'\x80\x04\x95\x16\x00\x00\x00\x00\x00\x00\x00}\x94\x8c\x03hmm\x94\x8c\tobjec
         t???\x94s.'
```

random

```
In [55]: import random

         random.randint(0, 100)
         random.random()

         a = [1, 2, 3, 4, 5]
         random.shuffle(a)
         a

         random.sample(a, 3)
         random.choices(a, k=10)

Out[55]: 75

Out[55]: 0.9875296991560779

Out[55]: [3, 2, 5, 1, 4]

Out[55]: [2, 3, 1]

Out[55]: [2, 1, 5, 1, 4, 2, 5, 4, 4, 1]
```

```
In [56]: from collections import Counter

         a = [1, 2, 3, 4, 5]
         weights = [5, 1, 1, 1, 1]

         Counter(random.choices(a, weights=weights, k=1000))

Out[56]: Counter({1: 594, 4: 109, 5: 92, 3: 100, 2: 105})
```

re

```
In [57]: import re

         comp = re.compile('\W+')
         res = re.split(comp, 'lal ??lala')
         res

         re.sub('aa', 'lol ', 'aaaaaa')

Out[57]: ['lal', 'lala']

Out[57]: 'lol lol lol '
```

```
In [58]: st = 'lldsldslajsjsjssajsjas'
         st2 = 'sslslssslsl??sswew'
         for s in (st, st2):
             res = re.fullmatch(re.compile('[a-z]*'), s)
             if res:
                 print('yes', s)
             else:
                 print('no', s)

         yes lldsldslajsjsjssajsjas
         no sslslssslsl??sswew
```

string

```
In [59]: import string

         string.punctuation
         string.ascii_lowercase
         string.ascii_uppercase

         string.printable
         string.whitespace

Out[59]: '!"#$%&\'()*+,-./:;<=>?@[\\]^_`{|}~'

Out[59]: 'abcdefghijklmnopqrstuvwxyz'

Out[59]: 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'

Out[59]: '0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ!"#$%&\'()
         *+,-./:;<=>?@[\\]^_`{|}~ \t\n\r\x0b\x0c'

Out[59]: ' \t\n\r\x0b\x0c'
```

```
In [60]: for letter in string.ascii_lowercase:
             print(letter, end=' ')

         a b c d e f g h i j k l m n o p q r s t u v w x y z
```

time

```
In [61]: import time

         time.time()
         time.asctime()
         time.localtime()
         time.timezone
         time.sleep(1)

Out[61]: 1586924097.9183745

Out[61]: 'Wed Apr 15 07:14:57 2020'

Out[61]: time.struct_time(tm_year=2020, tm_mon=4, tm_mday=15, tm_hour=7, tm_min=14, tm_s
         ec=57, tm_wday=2, tm_yday=106, tm_isdst=0)

Out[61]: -10800
```

typing

```
In [62]: import typing as tp

         def func(abc: str, l: tp.List[int]) -> tp.Dict[int, int]:
             str_len: int = len(abc)
             return dict(zip(l, l))

         func('a', [1, 2, 3])
         # for mypy

Out[62]: {1: 1, 2: 2, 3: 3}
```

uuid

```
In [63]: import uuid

         uuid.uuid4(), uuid.uuid4() # всегда дает рандомный uuid
         uuid.uuid1(), uuid.uuid1() # разные номера дают разные результаты в зависимости

Out[63]: (UUID('7ac5c9fc-ed59-4ce4-9b1f-ec2950218550'),
          UUID('595a6bcd-e48e-414e-a2b0-ab930c4dd90a'))

Out[63]: (UUID('accbc85c-7ecf-11ea-9ca6-00e04c0f68df'),
          UUID('accbc85d-7ecf-11ea-9ca6-00e04c0f68df'))
```