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

PYTHON 3

Introduction

Pavel Kositsyn

MIPT 2020

Как сдать курс

Активности:

1. Контесты (5-7)
   - 10 баллов за каждый
   - 2 недели на контест
2. Code Review (2)
   - Работа над кодом в **несколько** итераций
   - Есть обязательная часть
   - Есть бонусная часть: до 2 доп.баллов
   - 2 недели на первую (**рабочую!**) попытку
   - 1 неделя на каждую следующую итерацию
3. Лабораторная работа (1):
   - 10 баллов
   - Необходимо набрать хотя бы 1

1. Если вы сдали 2 ревью и лабораторную работу

$$\min(10, \lfloor 9 * \frac{\text{contest+lab}}{\text{max\_contest+max\_lab}} + \text{review\_bonus} \rfloor)$$

Например, 6 контестов + 1 лаба
max_points = 70

Кейс 1: Вы набрали 70 баллов и получили все бонусы за ревью:

$$\min(10, \lfloor 9 * \frac{70}{70} + 4 \rfloor) = \min(10, 13) = 10$$

Кейс 2: 35 баллов и по бонусному баллу за каждое ревью:

$$\lfloor 9 * \frac{35}{70} + 2 \rfloor = \lfloor 6.5 \rfloor = 6$$

Кейс 3: 24 балла и ни одного бонусного балла:

$$\lfloor 9 * \frac{24}{70} + 0 \rfloor = \lfloor 3.0857.. \rfloor = 3$$

Кейс 4: 23 балла и ни одного бонусного балла:

$$\lfloor 9 * \frac{23}{70} + 0 \rfloor = \lfloor 2.9571.. \rfloor = 2$$

2. Если вы сдали 1 ревью И лабораторную работу

Верхняя грань предыдущей по пятибалльной шкале оценки

Кейс 1:

$$\text{отл}(10) \rightarrow \text{хор}(7)$$

Кейс 2:

$$\text{хор}(6) \rightarrow \text{уд}(4)$$

Кейс 3:

$$\text{уд}(3) \rightarrow \text{неуд}(2)$$

Кейс 4:

$$\text{неуд}(2) \rightarrow \text{неуд}(2)$$

3. Если вы сдали 0 ревью или не сдали лабораторную работу (== набрали 0 баллов)

Intro

Язык Python разрабатывался Гвидо ван Россумом с 1989 г. Назван в честь популярного британского комедийного телешоу 1970-х "Летающий цирк Монти Пайтона"

Мажорные версии:

- Python 1.0 — январь 1994
- Python 2.0 — 16.10.2000
  - Python 2.7 - 3.07.2010
- Python 3.0 — 3.12.2008
  - Python 3.7.0 - 27.06.2018
  - Python 3.8.0 - 14.10.2019
  - Python 3.8.1 - 18.12.2019

Сейчас поддерживаются Python 2.x и Python 3.x:

- Синтаксически несовместимы
- **Поддержка 2.x прекратится в апреле 2020**

Python vs C++:

Интерпретируемость vs Компилируемость

- В компилируемых языках код программы нужно скомпилировать в т.н. исполняемый код. Этот код затем будет исполнять нечто, что ничего не знает про исходный язык программирования.
- В интерпретируемом же языке исполнителем является сама программа, которая анализирует исходный код. Python - интерпретируемый язык (и исполнителем кода на нём является программа под названием "python"). Из этого следуют несколько важных свойств:
  - Большинство ошибок возникает на этапе выполнения программы
  - Более гибкий и плотный код

Сильная динамическая типизация

- Сильная: У каждого объекта есть один конкретный тип; не бывает так, что мы не знаем, какого типа объект, или что объект относится к некоторому "неопределённому типу", который будет разрешён впоследствии, или что у нас есть объекты некого "универсального" типа.
- Динамическая: Мы *почти* никак не можем из исходного кода программы понять, какого типа объект в данном выражении. По крайней мере, не можем понять в точности. Чтобы это выяснить, нужно проинтерпретировать всю программу с самого начала до данного выражения.

Управление памятью

Не нужно следить за выделяемой памятью и чистить за собой: объекты удаляются тогда, когда на них нет ссылок. Тем не менее, забить память все еще можно.

Дзен Питона

In [2]:
```
import this
```

```
The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!
```

Как запустить что-то на Python

- Интерпретатор командной строки
- Запуск интерпретации файла
- Продвинутые интерпретаторы командной строки (ipython, ipython-notebook)
- Базовый редактор (idle)
- Продвинутые IDE (pycharm)

Основные типы

In [3]:
```
42000000000000000000000000000000000000000000000000
42.42
1 + 2.1j
True
"abc"
b'abc'
None
```

Out[3]: 42000000000000000000000000000000000000000000000000

Out[3]: 42.42

Out[3]: (1+2.1j)

Out[3]: True

Out[3]: 'abc'

Out[3]: b'abc'

In [4]:
```
type(42)
type(42.42)
type(1 + 2.1j)
type(True)
type('abc')
type(b'abc')
type(None)
id(int)
```

Out[4]:
```
int
```

Out[4]:
```
float
```

Out[4]:
```
complex
```

Out[4]:
```
bool
```

Out[4]:
```
str
```

Out[4]:
```
bytes
```

Out[4]:
```
NoneType
```

Out[4]:
```
11291776
```

Арифметика

In [5]:
```
(1 + 6) / 4
45 // 0.2
45 % 6
623903980938403938402938402938 * 232982349823094820934802938 4
123456789 ** 12345
1 ^ 1
```

Out[5]:
```
1.75
```

Out[5]:
```
224.0
```

Out[5]:
```
3
```

Out[5]:
```
14535861554301397808581975386405822962036044853167169512
```

Out[5]:
```
565448628007380331029018059495140595085075647769981877555601786144208288268375145368853797742347184495141604702655893464499016600093315596796792846438319858294284934753797953202267698829387845045100784956246981838243105294709242966838699050071136253846845623900492430045595300396273280088530629410940611182861568301038902823806554037719290503535259209722815774068231863326881596671663543098267439465838898495241998917679873311256402142872977815875519737936382837924501350933111804913170562349202716343276535318155943909659121452890695847976611733257135517471848082616342510243123169715343633526414799980312308095721623830931874121705209340930359796008603496626652567618480663660540962071281587227704600682863313555927612297725240398285952908479117174370914232819398915847268088508595199686863816044392213774786732337935040743089028778214442466814382009409329230354834958888919120470165255505055056029042514301744060517844126648383518424403321573648591545256818308187090101332662752988799470599539791564223698030408188269945585 1
```

Оператор связывания

In [6]:
```
a = 10000
b = a
a is b
id(a)
id(b)
```

Out[6]: True

Out[6]: 139757904871856

Out[6]: 139757904871856

In [7]:
```
a = 10000
b = 10000
a is b
id(a)
id(b)
```

Out[7]: False

Out[7]: 139757904872688

Out[7]: 139757904873072

In [8]:
```
a = 123
b = 123
a is b
id(a)
id(b)
# Пример того, что числа до 256 - это синглтон-объекты, "вшитые" в окружение ис
```

Out[8]: True

Out[8]: 11500544

Out[8]: 11500544

In [9]:
```
a = 1000
b = 1010
b -= 10
id(a) == id(b) # a is b
```

Out[9]: False

Контейнеры

In [10]:
```
import sys

s = 'ab'
sys.getsizeof(s)

d = 'abc'
sys.getsizeof(d)
```

Out[10]: 51

Out[10]: 52

In [11]:
```
s = 'abcdef'                # str       i
l = [1, 2, 2.5, 'abc']      # list      m
t = (42, 41, True)          # tuple     i
S = {1, 2, 3}               # set       m
fs = frozenset({1, 2, 3})   # frozenset i
```

In [12]:
```
type(s), sys.getsizeof(s)
type(l), sys.getsizeof(l)
type(t), sys.getsizeof(t)
type(S), sys.getsizeof(S)
type(fs), sys.getsizeof(fs)
```

Out[12]: (str, 55)

Out[12]: (list, 88)

Out[12]: (tuple, 64)

Out[12]: (set, 216)

Out[12]: (frozenset, 216)

In [13]:
```
a = (1 << 30) - 1
sys.getsizeof(a)
```

Out[13]: 28

In [14]:
```
# List
l[1] = 378272
l[-1] = 'cba'
l

len(l)
```

Out[14]: [1, 378272, 2.5, 'cba']

Out[14]: 4

In [15]:
```
l = [1, 2, 3]
l.append(4)
# l = [1, 2, 3, 4]

l.pop()
# l = [1, 2, 3]

l.pop(0) # работает за линию
# l = [2, 3]

l.index(2) # тоже линия
l

5 in l # аналогично

l.index(5)
```

Out[15]: 4

Out[15]: 1

Out[15]: 0

Out[15]: [2, 3]

Out[15]: False

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-15-c94964db7217> in <module>
     14 5 in l # аналогично
     15
---> 16 l.index(5)

ValueError: 5 is not in list
```

In [16]:
```
t = ('i', 'cannot', 'be', 'modified')
t[1] = 'can'
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-16-0ac23e849d51> in <module>
      1 t = ('i', 'cannot', 'be', 'modified')
----> 2 t[1] = 'can'

TypeError: 'tuple' object does not support item assignment
```

In [17]:
```
t = ('i', ['cannot'], 'be', 'modified')
t[1][0] = 'actually can sometimes'
t
```

Out[17]:
```
('i', ['actually can sometimes'], 'be', 'modified')
```

In [18]:
```
u = ()
type(u)
```

Out[18]:
```
tuple
```

In [19]:
```
# Множество
{3, 3, 3, 1, 1, 1, 2, 2, 2, 3, 3, 3}

# А это множество?
type({})

type(set())
```

Out[19]:
```
{1, 2, 3}
```

Out[19]:
```
dict
```

Out[19]:
```
set
```

In [20]:
```
s = set()
s.add(2) # O(1)
# s = {2}

d = {1, 2, 3}
d.remove(1) # O(1)
# d = {2, 3}

s | d, s & d, s ^ d

1 in s  # O(1)
2 in s
```

Out[20]:
```
({2, 3}, {2}, {3})
```

Out[20]:
```
False
```

Out[20]:
```
True
```

Арифметика с разными типами

In [21]:
```
'Hello ' + 'world'
```

Out[21]:
```
'Hello world'
```

In [22]:
```
'abc' * 4
```

Out[22]:
```
'abcabcabcabc'
```

In [23]:
```
'abc'[0]
'abc'[0] = 'd'
```

Out[23]:
```
'a'

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-23-d899994c2803> in <module>
      1 'abc'[0]
----> 2 'abc'[0] = 'd'

TypeError: 'str' object does not support item assignment
```

In [24]:
```
'Thats how you get a list of words'.split() # можно в агрументах дать свой сепар

'split by y letter???'.split('y')
'y'.split('y')
```

Out[24]:
```
['Thats', 'how', 'you', 'get', 'a', 'list', 'of', 'words']
```

Out[24]:
```
['split b', ' ', ' letter???']
```

Out[24]:
```
['', '']
```

In [25]:
```
[1] + [2, 3]

lst0 = [1]
lst0.append([2, 3])
lst0

lst = [1]
lst.append(lst)
lst[1][1][1][1][1][1][1]
```

Out[25]:
```
[1, 2, 3]
```

Out[25]:
```
[1, [2, 3]]
```

Out[25]:
```
[1, [...]]
```

In [26]:
```
[2, 3, 4] * 3
```

Out[26]:
```
[2, 3, 4, 2, 3, 4, 2, 3, 4]
```

In [27]:
```
[[]] * 3

lst = [[]] * 3
lst[0].append(1)
lst
```

Out[27]:
```
[[], [], []]
```

Out[27]:
```
[[1], [1], [1]]
```

In [28]:
```
[1] + 2
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-28-54324b7054ad> in <module>
----> 1 [1] + 2

TypeError: can only concatenate list (not "int") to list
```

Приведение типов

In [29]:
```
int('123')

int('102', 3)
```

Out[29]:
```
123
```

Out[29]:
```
11
```

In [30]:
```
str(345)
```

Out[30]:
```
'345'
```

In [31]:
```
x = 5
float(x)

float('-inf')
d = -inf
```

Out[31]:
```
5.0
```

Out[31]:
```
-inf
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-31-29485cbaad37> in <module>
      3
      4 float('-inf')
----> 5 d = -inf

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

In [32]:
```
int(234.5)

# оффтоп
round(234.5)
round(234.50001)
round(234.5123, 2)
```

Out[32]:
```
234
```

Out[32]:
```
234
```

Out[32]:
```
235
```

Out[32]:
```
234.51
```

In [33]:
```
int('12a', 16)
```

Out[33]:
```
298
```

In [34]:
```
l = [1, 2, 1, 5, 5, 6]
set(l)
```

Out[34]:
```
{1, 2, 5, 6}
```

In [35]:
```
t = (1, [])
l = list(t)
l[1] = [2]
t, l
```

Out[35]:
```
((1, []), [1, [2]])
```

In [36]:
```
s = "abab"
tuple(s)
```

Out[36]:
```
('a', 'b', 'a', 'b')
```

In [37]:
```
list(True)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-37-9fe58a8fbfd7> in <module>
----> 1 list(True)

TypeError: 'bool' object is not iterable
```

Приведение к bool

In [38]:
```
bool(0)
bool(1)
bool(-2)
```

Out[38]:
```
False
```

Out[38]:
```
True
```

Out[38]:
```
True
```

In [39]:
```
bool(0.0)
bool(-12.3)
bool(24.1)
```

Out[39]:
```
False
```

Out[39]:
```
True
```

Out[39]:
```
True
```

In [40]:
```
bool('')
bool('abc')
```

Out[40]:
```
False
```

Out[40]:
```
True
```

In [41]:
```
bool((1, 2))
bool(())
```

Out[41]:
```
True
```

Out[41]:
```
False
```

In [42]:
```
bool([None])
bool([])
```

Out[42]: True

Out[42]: False

In [43]:
```
bool({12.5, 45.7})
bool(set())
```

Out[43]: True

Out[43]: False

In [44]:
```
bool(None)
```

Out[44]: False

Сравнения

In [45]:
```
3 > 1
```

Out[45]: True

In [46]:
```
2 <= 5
```

Out[46]: True

In [47]:
```
1 == -1
```

Out[47]: False

In [48]:
```
1 != -1
```

Out[48]: True

In [49]:
```
'a' > 'b'  # строки сравниваются лексикографически: по номеру буквы в алфавите
```

Out[49]: False

In [50]:
```
'aaa' > 'ab'  # буквы сравниваются по очереди
```

Out[50]: False

In [51]:
```
[1, 2, 3] == [2, 1, 3]

{1, 2} == {2, 1}

id(1000) == id(1000)
type(1000) == type(10)
```

Out[51]: False

Out[51]: True

Out[51]: True

Out[51]: True

Булева алгебра

A	B	A or B	A and B
False	False	False	False
False	True	True	False
True	False	True	False
True	True	True	True

In [52]:
```
True and False
```

Out[52]: False

In [53]:
```
True or False
```

Out[53]: True

In [54]:
```
not True
```

Out[54]: False

Применимо и к объектам!

bool(A)	bool(B)	A or B	A and B
False	-	B	A
True	-	A	B

In [55]:
```
[2] and [1]
```

Out[55]: [1]

In [56]:
```
'' or 'spam'
```

Out[56]: 'spam'

In [57]:
```
None and True
```

In [58]:
```
None or False
```

Out[58]: False

In [59]:
```
import random

coinflip = random.randint(0, 1)
if coinflip:
    l = [1]
else:
    l = None

l = l or [] # Избегаем None
l
```

Out[59]: []

In [60]:
```
cond = None

if cond == None:
    True

if cond is None:
    True

if not cond:
    True
```

Out[60]: True

Out[60]: True

Out[60]: True

Распаковка

In [61]:
```
a, b, c = (2, 3, 4)
a, b, c
```

Out[61]: (2, 3, 4)

In [62]:
```
# swap
a = 3
b = 4
a, b = b, a

a, b
```

Out[62]: (4, 3)

Ввод-вывод

In [63]:
```
input()
```

```
lala
```

Out[63]: 'lala'

In [64]:
```
int(input())
```

```
10
```

Out[64]: 10

In [65]:
```
list(map(int, input().split())) # магия
```

```
1 2 3 4
```

Out[65]: [1, 2, 3, 4]

In [66]:
```
print(345, 100, sep='\t', end=' ')
print('old line')
```

```
345     100 old line
```