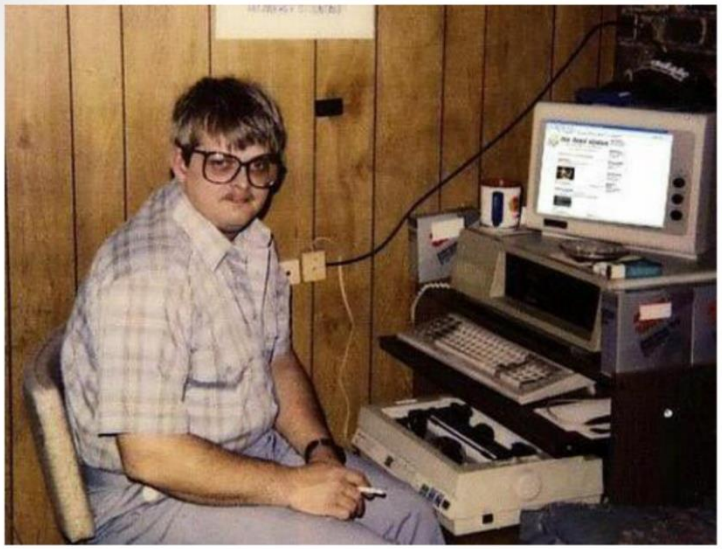
PYTHON

ВВЕДЕНИЕ

ПОЧЕМУ PYTHON?

- Прост в изучении
- Большое количество модулей
- Простые конструкции
- …

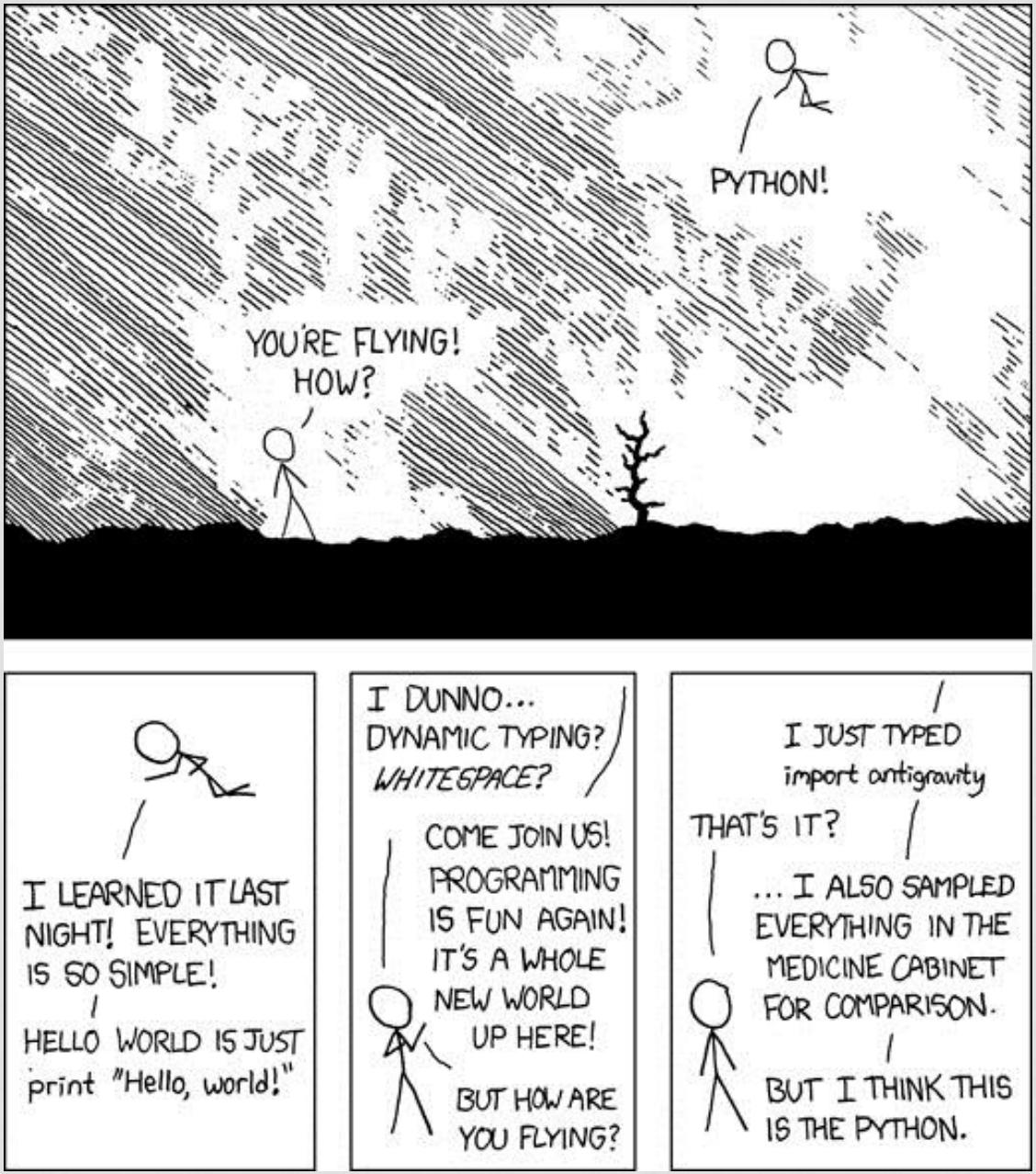
HOLY WAR!



Программист на Perl смотрит на тебя как на $_[0]

Программист на Haskell поднимает штангу именем теории категорий

Ruby-программисты – заботливые отцы, а значит их любят девушки


PYTHON!
YOU'RE FLYING! HOW?
I LEARNED IT LAST NIGHT! EVERYTHING IS SO SIMPLE!
HELLO WORLD IS JUST print "Hello, world!"
I DUNNO... DYNAMIC TYPING? WHITESPACE?
COME JOIN US! PROGRAMMING IS FUN AGAIN! IT'S A WHOLE NEW WORLD UP HERE!
BUT HOW ARE YOU FLYING?
I JUST TYPED import antigravity
THAT'S IT?
... I ALSO SAMPLED EVERYTHING IN THE MEDICINE CABINET FOR COMPARISON.
BUT I THINK THIS IS THE PYTHON.

PYTHON

Python - высокоуровневый язык программирования общего назначения с акцентом на производительность разработчика и читаемость кода. Синтаксис ядра Python минималистичен.

Python поддерживает несколько парадигм программирования, в том числе структурное, объектно-ориентированное, функциональное, императивное и аспектно-ориентированное. Основные архитектурные черты — динамическая типизация, автоматическое управление памятью, полная интроспекция, механизм обработки исключений, поддержка многопоточных вычислений.

PYTHON - ДЗЕН

- >>> 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 ВЫПОЛНЯТЬ

- Cpython
- IronPython - .NET
- Jython - Интерпретатор Python, реализованный на Java. Позволяет компилировать программы на Python в байт-код Java.
- PyPy
- ….

ЗАПУСК

- python
- #!/usr/bin/env python

  …
- IPython

PYTHON 2.X VS 3.X

- В третьей версии улучшена стандартная библиотека и добавлены новые функции.
- Много библиотек не стабильно работают на версии 3.

- На данный момент 2.7.5 vs 3.3.2

ТИПЫ ДАННЫХ

- Int
- “long int”
- float
- complex

```
>> 4j + 2 + 3j
(2+7j)
>> complex (2,7)
(2+7j)
>> (2+7j ). real + (2+7j ). imag
9.0
>> (2+7j ). conjugate ()
(2-7j)
```

ПЕРЕМЕННЫЕ

```
>>> x = 2
>>> x
2
>>> print(x)
2
```

ПОЛУЧЕНИЕ ДАННЫХ ОТ ПОЛЬЗОВАТЕЛЯ

```
>>> x = input("Hello: ")
Hello: aaaa
>>> x
'aaaa'
>>> x = int(input("Hello: "))
Hello: 123
>>> x
123
```

УСЛОВИЯ

```
>>> if x < 0:
...     x = 0
...     Print(“Negative”)
... elif x == 0:
...     Print(“Zero”)
... else:
...     Print(“Positive”)
```

Сравнение: == != >= <=

x = a if (condition) else b

WHILE

```
>>> #comment
... a, b = 0, 1 # множественное присваивание
>>> while b < 60:
...     print(b)
...     a, b = b, a+b
...
1
1
2
3
5
8
13
21
34
55
```

WHILE

```
>>> #comment
... a, b = 0, 1
>>> while b < 100:
...     print(b, end=“”)
...     a, b = b, a+b
...
1 1 2 3 5 8 13 21 34 55 89
```

КОММЕНТАРИИ

- однострочный комментарий
“““ – многострочный комментарий. Доступен через __doc__ или help(…)

```
>>> def sum(a, b):
...     """
...     Returns sum of a and b
...     """
...     return a+b
>>> print sum.__doc__

    Returns sum of a and b

>>> help(sum)
```

СПИСКИ

```
>>> a = ['spam', 'eggs', 100, 1234]
>>> a[2] = a[2] + 23
>>> a
['spam', 'eggs', 123, 1234]
```

Присваивание срезу:

```
>>> a[0:2] = [1, 12] # замена
>>> a
[1, 12, 123, 1234]
>>> a[0:2] = [] # удаление
>>> a
[123, 1234]
>>> a[1:1] = ['bletch', 'xyzzy'] # вставка
>>> a
[123, 'bletch', 'xyzzy', 1234]
```

СПИСКИ

Четыре способа добавить элементы в список.

```
>>> a_list = ['a']
>>> a_list = a_list + [2.0, 3]
>>> a_list
['a', 2.0, 3]
>>> a_list.append(True)
>>> a_list
['a', 2.0, 3, True]
>>> a_list.extend(['four', 'Ω'])
>>> a_list
['a', 2.0, 3, True, 'four', 'Ω']
>>> a_list.insert(0, 'Ω')
>>> a_list
['Ω', 'a', 2.0, 3, True, 'four', 'Ω']
```

СПИСКИ

Удаление элементов из списка:.

```
>>> a_list = ['a', 'b', 'new', 'mpilgrim', 'new']
>>> del a_list[1]
>>> a_list
>>> ['a', 'new', 'mpilgrim', 'new']
>>> a_list.remove('new')
>>> a_list
['a', 'mpilgrim', 'new']
>>> a_list.pop()
'new'
>>> a_list
['a', 'mpilgrim']
>>> a_list.pop(0)
‘a'
```

СПИСКИ

```
>>>a_list.remove(334)
Traceback (most recent call last):
  File "<pyshell#84>", line 1, in <module>
    a_list.remove(334)
ValueError: list.remove(x): x not in list
```

СПИСКИ. ПОИСК

```
>>> a_list = ['a', 'b', 'new', 'mpilgrim', 'new']
>>> a_list.count('new')
2
>>> 'new' in a_list
True
>>> a_list.index('mpilgrim')
3
>>> a_list.index('new')
2
>>> a_list.index('c')
Traceback (innermost last):
File "<interactive input>", line 1, in ?
ValueError: list.index(x): x not in list
```

КОРТЕЖИ

Значения менять нельзя!

```
>>> t = 12345, 54321, 'hello!'
>>> t[0]
12345
>>> t
(12345, 54321, 'hello!')
>>> u = t, (1, 2, 3, 4, 5) # могут быть вложенными
>>> u
((12345, 54321, 'hello!'), (1, 2, 3, 4, 5))
>>> empty = ()
```

FOR, RANGE

```
>>> list = [’It’, ’is an’, ’interesting’, 'lecture']
>>> for x in list:
. . . print(x, end=“”)
. . .
It is an interesting lecture
для удобства
>>> range(10)
#[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>> range(5, 10) # диапазон
#[5, 6, 7, 8, 9]
>>> range(0, 10, 3) # задаем шаг
#[0, 3, 6, 9]
```

СЛОВАРИ

```
>>> dict = {} # пустой словарь
>>> circus = {“lion” : 4, “hippo” : 1, “giraffe” : 2}
>>> circus[“hippo”]
1
>>> circus[“snake] = 7# добавление ключа
>>> circus[“lion”] = 5# изменение ключа
>>> circus
{‘hippo’: 1, ‘lion’: 5, ‘giraffe’: 2, ‘snake’: 7}
```

СЛОВАРИ

```
>>> len(circus)#количество элементов в словаре
5
>>> circus[“cat”] = “yes, please!”#разные типы значений
>>> circus[42] = “number”# разные типы ключей
>>> del circus[“hippo”]# удаление ключа
>>> circus
{42: ‘number’, ‘cat’: ‘yes, please!’, ‘lion’: 5, ‘giraffe’: 2, ‘snake’:
  7}
>>> circus.keys()# возвращает список ключей
[42, ‘cat’, ‘lion’, ‘giraffe’, ‘snake’]
>>> circus.values()# возвращает список значений
[‘number’, ‘yes, please!’, 5, 2, 7]
>>> ‘dog’ in circus#проверяет наличие ключа в словаре
False
```

При обращении по несуществующему ключу – исключение KeyError

ИСКЛЮЧЕНИЯ

```
>>> print(circus[‘dog’])
Traceback(most recent call last):
 File "<pyshell#1>", line 1, in <module>
  print circus['dog']
KeyError: 'dog‘
```

Можно поймать это исключение:

```
try:
   print circus['dog']
except KeyError:
   print "No such key in the dictionary"
```

ФУНКЦИИ

Передача аргументов в функцию - по ссылке.

```
def dostuff(mylist) :
    “““ Appends [1, 2, 3] to the list. ”””
    mylist.append([1, 2, 3])
    mylist= [‘Katya’, ‘Kolya’, ‘Vitya’]
    mylist.append(‘Sveta’)
    print(mylist)
a = [4, 5, “f”]
dostuff(a)# результат: [‘Katya’, ‘Kolya’, ‘Vitya’, ‘Sveta’]
print(a)# результат: [4, 5, ‘f’, [1, 2, 3]]
```

ФУНКЦИИ

НО!

```
def square(n) :
  n *= n
a = 3
square(a)
print a
```

Результат выполнения:

3

Потому что число – простой тип, оно передается по значению.

ПЕРЕМЕННЫЕ

Переменные внутри функции –локальные. Поиск переменных: сперва среди локальных, потом среди глобальных, потом среди встроенных.

```
n = 10
def printn():
    print(n)# переменная n видна внутри функции
def changeandprintn()
    n = 2# теперь n –это локальная переменная
    print(n)
printn()
changeandprintn()
print n
```

Результат выполнения:

```
10
2
10
```

ПЕРЕМЕННЫЕ

```
n = 10
def reallychangeandprintn()
    global n# переменная n –глобальная
    n = 2
    print n
reallychangeandprintn()
print n
```

Результат:

2

2

global a, b, c-указывает, что идентификаторы a, b, c в текущем блоке ссылаются на глобальные переменные

ФУНКЦИИ

Как передать в функцию произвольное число аргументов:

f([formal_args,] *tuple): tuple–кортеж, содержащий аргументы, не входящие в список формальных параметров

```
def mean(*args):
    sum = 0.
    for a in args:
        sum += a
    return sum/len(args)
print mean(1, 2, 3, 4, 5)# результат: 3
print mean(40, 3)# результат: 21.5
```

ПРО ВСТРОЕННЫЕ ТИПЫ

False = None, 0, 0.0, 0j, ‘’, (), [], {}

Boolean operations:

- X or Y
- X and Y
- not X

Numeric types

- x+y, x-y, x*y, x/y, -x, +x
- x//y, x%y, pow(x, y), x**y
- math.trunc(x)
- round(x [, n])
- math.floor(x), math.ceil(x)

ЗАДАНИЕ

Необязательные:

1. **http://www.pythonchallenge.com/**
2. Посчитать количество существительных в романе «Война и мир». Смотреть на **Pymorphy**

Обязательные

3. Найдите все составные числа меньшие N, которые представимы в виде произведения двух простых чисел. Число N пользователь вводит с клавиатуры при запуске.
4. Написать функцию, вычисляющую произведение двух матриц (матрица – список списков). Если матрицы нельзя перемножить, сгенерировать исключение. Также написать функцию для вывода такой матрицы в красивом виде.