

# Знакомство с языком Python



# План

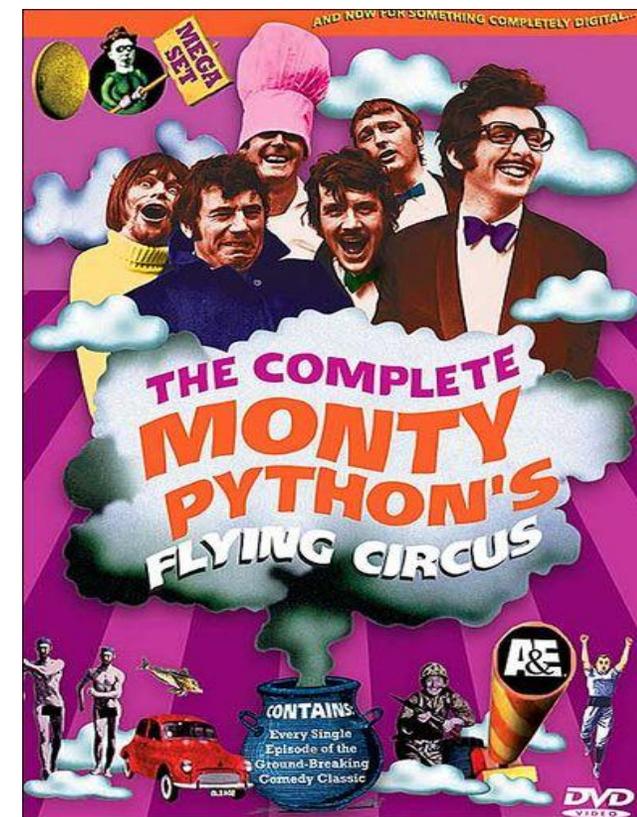
- История создания, философия
- Зачем нужен, где используют
- Примеры кода Python
- Примеры простеньких игр, написанных мной с помощью Python

# История создания

- Python был разработан в конце 1989 г.

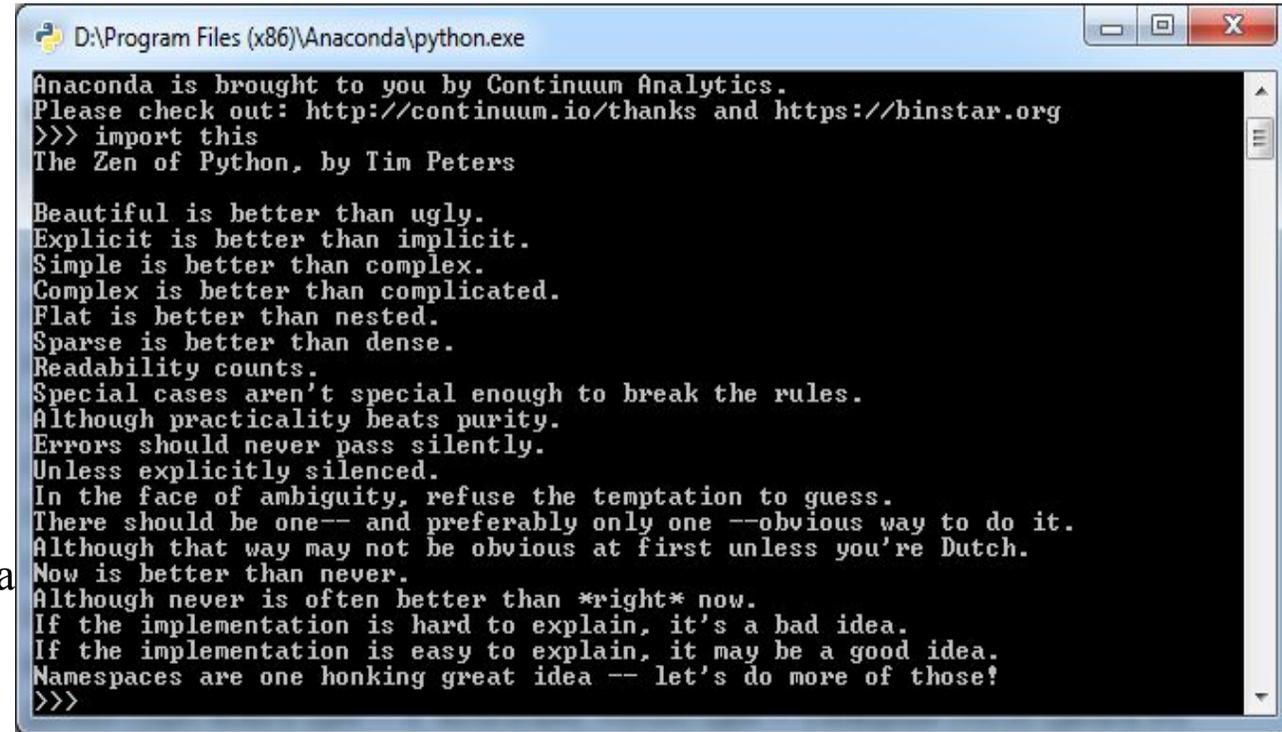
Гuido ван Россумом

- Автор назвал язык в честь популярного британского комедийного телешоу 1970-х «Летающий цирк Монти Пайтона». Но чаще ассоциируют со змеей.



# Философия Python

- Красивое лучше, чем уродливое
- Явное лучше, чем неявное
- Простое лучше, чем сложное
- Сложное лучше, чем запутанное
- Плоское лучше, чем вложенное
- Разреженное лучше, чем плотное
- Читаемость имеет значение
- Особые случаи не настолько особые, чтобы нарушать правила
- При этом практичность важнее безупречности
- Ошибки никогда не должны замалчиваться
- Если не замалчиваются явно
- Встретив двусмысленность, отбрось искушение угадать
- Должен существовать один — и, желательно, только один — очевидный способ сделать это
- Хотя он поначалу может быть и не очевиден, если вы не голландец
- Сейчас лучше, чем никогда
- Хотя никогда зачастую лучше, чем прямо сейчас
- Если реализацию сложно объяснить — идея плоха
- Если реализацию легко объяснить — идея, возможно, хороша
- Пространства имён — отличная штука! Будем делать их побольше!



```
D:\Program Files (x86)\Anaconda\python.exe
Anaconda is brought to you by Continuum Analytics.
Please check out: http://continuum.io/thanks and https://binstar.org
>>> 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!
>>>
```

# Python - ориентирован на разработчика

- Программы на Python легко читаются, лаконичны
- Большая стандартная библиотека
- Работа с сетью, web, работа с файлами баз данных, архивами, мультипоточность, мультипроцессорность, высокоуровневые структуры данных (комплексные числа, списки, словари, множества)
- Подходит для быстрой разработки программ, прототипирования

# Python — масштабируемый , интерпретируемый

Масштабируемость кода:

- наборы команд объединяются в функции
- функции объектов объединяются в классы
- наборы функций и классов объединяются в модули(отдельные файлы)
- модули группируются в пакеты (директории с файлами модулей)

Масштабируемость по производительности:

- узкие места программ можно переписать на C или C++

# Применение Python

- Список компаний, которые используют Python, длинный. Среди них Google, Facebook, Yahoo, NASA, RedHat, IBM, Instagram, Dropbox, Яндекс, Mail.Ru
- BitTorrent (все версии до 6 этого торрент-клиента были написаны на Python. Версия 6 была переписана на C++)
- Ubuntu Software Center
- Blender( Python используется как средство создания инструментов и прототипов, системы логики в играх, как средство импорта/экспорта файлов (например COLLADA), автоматизации задач)
- GIMP (Python используется для написания дополнительных модулей, например, фильтров)
- Игры, Civilization IV (Большая часть игры написана на Python), Battlefield 2 (в сети есть много учебников и просто рецептов по изменению различных объектов и их поведения), World of Tanks.

# Конструкции Python

```
CodeSkulptor
```

```
1 print "Hello world"
```

Hello world

```
CodeSkulptor
```

```
1 # For loop on a list
2 numbers = [2, 4, 6, 8]
3 product = 1
4 for number in numbers:
5     product = product * number
6
7 print('The product is:', product)
```

('The product is:', 384)

```
CodeSkulptor
```

```
1 # Python 3: List comprehensions
2 fruits = ['Banana', 'Apple', 'Lime']
3 loud_fruits = [fruit.upper() for fruit in fruits]
4 print(loud_fruits)
5
6 # List and the enumerate function
7 print list(enumerate(fruits))
```

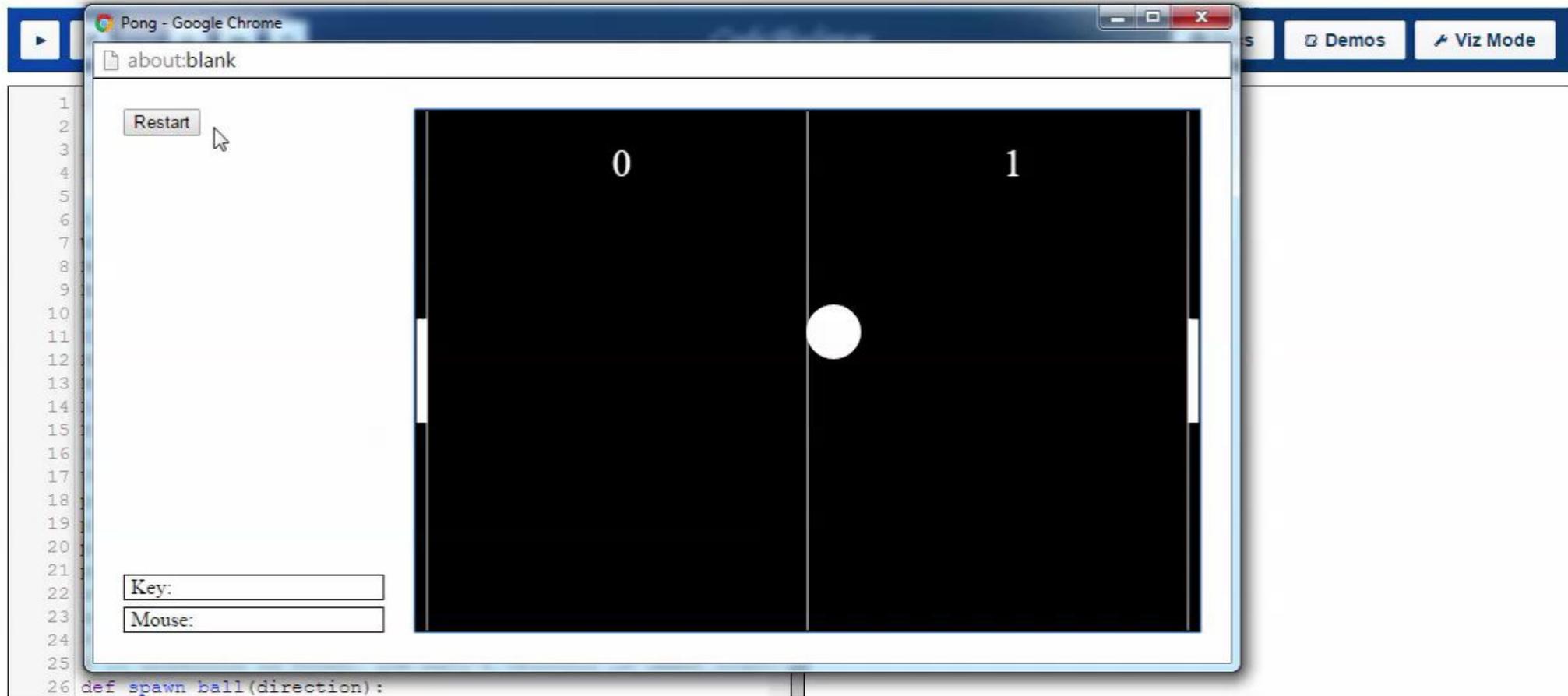
['BANANA', 'APPLE', 'LIME']  
[(0, 'Banana'), (1, 'Apple'), (2, 'Lime')]

```
CodeSkulptor
```

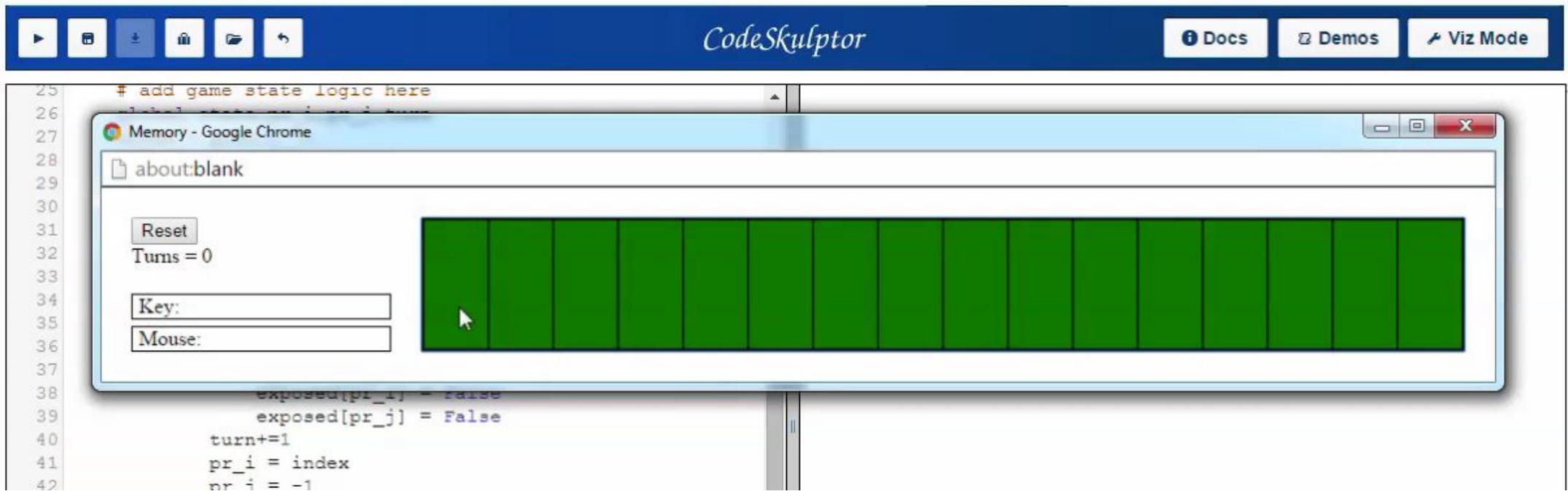
```
1 # Python 3: Simple arithmetic
2 print 1 / 2
3 print 2 ** 3
4
5 print 17. / 3 # classic division returns a float
6
7 print 17 // 3 # floor division
8
```

0  
8  
5.666666666667  
5.0

# ПОНГ



# Память



The image shows a CodeSkulptor interface with a browser window titled "Memory - Google Chrome" open. The browser window displays a game interface with a "Reset" button, "Turns = 0", a "Key:" input field, and a "Mouse:" input field. To the right of these controls is a horizontal row of 12 green rectangular cards. The CodeSkulptor code editor shows the following code:

```
25 # add game state logic here
26 label_state and turn
27
28
29
30
31
32
33
34
35
36
37
38 exposed[pr_i] = raise
39 exposed[pr_j] = False
40 turn+=1
41 pr_i = index
42 pr_i = -1
```