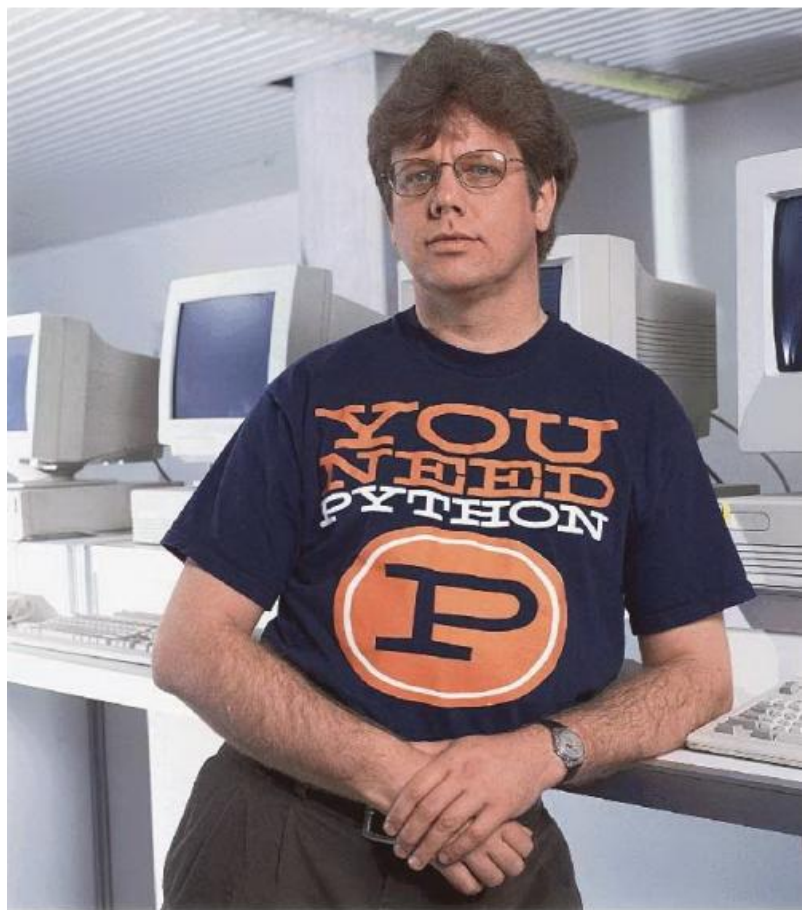


# Прошел месяц!

Удалось ли знакомство с Python?

# Создатель великий Гвидо



# Дзен питона или 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!
```

- 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 они 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 [<sup>1</sup>].
- 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 больше!

# Python и математика

- Привычные операции в привычном виде
- Возможность сравнивать числа
- Длинная арифметика
- `Import math`

# Python и строки

- Двойные и одинарные кавычки
- `Print(your string)`
- Возможность индексирования строк, взятия срезов
- Строки поддерживают итерирование
- Сложение строк
- Умножение строки на число

# Еще типы данных в Python

- Вещественные числа (float)
- Логический тип (bool)
- Строки (str)
- Целые числа (int)
- type(object)

# Структуры данных в Python

- Список (`type == list`)
- Кортеж (`type == tuple`)
- Множество (`type == set`)
- Словарь (`type == dict`)

# Условные операторы в Python

```
In [17]: if True:
...:     print("hello")
...: elif True:
...:     print("true")
...: else:
...:     print("bye")
...:
```



# Циклы в Python

## For

```
In [22]: for e in range(0, 10):  
.....:     print(e*e)  
.....: else:  
.....:     print("end")  
.....:
```

# Циклы в Python

```
In [18]: e = 0

In [19]: while e < 10:
.....:     print("hello")
.....:     e+=1
.....: else:
.....:     print("bye")
.....:
```

# ФУНКЦИИ В Python

- def
- args
- Обобщенная логика

```
In [24]: def func_hello_someone(name):  
        ...:     print(f"hello, {name}")  
        ...:  
  
In [25]: func_hello_someone("Misha Yudin")  
hello, Misha Yudin
```

# Анонимные функции в Python

- lambda

```
In [26]: f = lambda x: print(x*x)
```

```
In [27]: f(8)
```

```
64
```

# Модули или «батарейки»

- Import
- Import math, time, sys, this
- pip install <name of module>