**Python общие вопроси**

План

0. Источник

1. Особенности язика Python

2. Дзен Python

3. IDEA для разработки

4. Первая программа (Сode)

5. Интерпритатор Python (Интервью)

6. Анализ Traceback

7. PEP 8 - руководство по написанию кода на Python

0. Источник

<https://ru.wikipedia.org/wiki/Python>

<https://habr.com/ru/company/piter/blog/526718/>

<https://www.youtube.com/watch?v=oiHc0fryZ9c&list=RDCMUCIyfaiKil5oomY64XIRJEAA&index=25&ab_channel=HillelITSchool>

<https://www.youtube.com/watch?v=AkvVRUKWogk&list=RDCMUCIyfaiKil5oomY64XIRJEAA&start_radio=1&t=1634&ab_channel=HillelITSchool>

<https://www.youtube.com/watch?v=hgI0p1zf31k&feature=emb_logo&ab_channel=PythonDiscord>

<https://www.python.org/downloads/windows/>

1. Особенности язика Python

Python

* [мультипарадигмальный](https://ru.wikipedia.org/wiki/Мультипарадигмальный_язык_программирования);
* [объектно-ориентированное](https://ru.wikipedia.org/wiki/Объектно-ориентированное_программирование);
* [структурное программирование](https://ru.wikipedia.org/wiki/Структурное_программирование);
* ф[ункциональное](https://ru.wikipedia.org/wiki/Функциональное_программирование);
* [аспектно-ориентированное](https://ru.wikipedia.org/wiki/Аспектно-ориентированное_программирование);
* [метапрограммирование](https://ru.wikipedia.org/wiki/Метапрограммирование);
* [контрактное](https://ru.wikipedia.org/wiki/Контрактное_программирование) и [логическое программирование](https://ru.wikipedia.org/wiki/Логическое_программирование) можно реализовать с помощью расширений.

Язык использует

* [динамическую типизацию](https://ru.wikipedia.org/wiki/Динамическая_типизация) вместе с [подсчётом ссылок](https://ru.wikipedia.org/wiki/Подсчёт_ссылок)
* циклический [сборщик мусора](https://ru.wikipedia.org/wiki/Сборщик_мусора) для [менеджмента памяти](https://ru.wikipedia.org/wiki/Менеджер_памяти)
* динамические [разрешения имен](https://ru.wikipedia.org/w/index.php?title=Разрешения_имен_(программирование)&action=edit&redlink=1) ([динамическое связывание](https://ru.wikipedia.org/wiki/Динамическое_связывание)), которые связывают имена методов и переменных во время выполнения программы.

# Преимущества и недостатки языка Python (интервью)

<https://blog.ithillel.ua/articles/preimushchestva-i-nedostatki-yazyka-python>

# Python: сферы применения

<https://blog.ithillel.ua/videos/python-sfery-primeneniya>

2. Дзен Python

<https://pythonchik.ru/osnovy/dzen-python-pep20>

* Красивое лучше уродливого.
* Явное лучше неявного.
* Простое лучше сложного.
* Сложное лучше запутанного.
* Развернутое лучше вложенного.
* Разреженное лучше плотного.
* Читаемость имеет значение.
* Особые случаи не настолько особые, чтобы нарушать правила.
* При этом практичность важнее безупречности.
* Ошибки не должны замалчиваться.
* Если не замалчиваются явно.
* Встретив двусмысленность, отбрось искушение угадать.
* Должен существовать один - и, желательно, только один – очевидный способ сделать что-то.
* Хотя этот способ поначалу может быть и не очевиден, если вы не голландец.
* Сейчас лучше, чем никогда.
* Хотя никогда часто лучше, чем \*прямо\* сейчас.
* Если реализацию сложно объяснить – идея точно плоха.
* Если реализацию легко объяснить – возможно, идея хороша.
* Пространства имен – отличная штука! Будем использовать их чаще!

## Общие принципи програмирования:

## DRY;

## KISS

## YAGNI

3. IDEA для разработки

|  |  |
| --- | --- |
|  |  |
| [Ferrari SF90 Stradale](https://www.google.com/url?sa=i&url=https%3A%2F%2Fuk.wikipedia.org%2Fwiki%2FFerrari_SF90_Stradale&psig=AOvVaw0_dEhsxDjL4Mb57eotTKWi&ust=1612334012287000&source=images&cd=vfe&ved=0CA0QjhxqFwoTCODystPLyu4CFQAAAAAdAAAAABAD)  PyCharm | [Dodge Viper Challenger](https://www.motorious.com/articles/news/dodge-viper-challenger/)  VSCode |

Инструменти IDEA

1. Линтери;

2. Снипети;

3. Ранери.

4. Первая программа (Сode)

5. Интерпритатор Python

# [Погружение в пучину интерпретатора Python. Ч1](https://habr.com/ru/post/264609/)

# <https://habr.com/ru/post/264609/>

# Интерпритатор работает в четыре шага (интервью):

# 1. лексической анализ;

# 2. парсинг - Парсер забирает скормленные ему инструкции и генерирует структуру которая объясняет их связь формируя [AST](https://ru.wikipedia.org/wiki/Абстрактное_синтаксическое_дерево)( Абстрактное Синтаксическое Дерево);

# 3. компиляция - Компилятор затем преобразует AST в одни (или несколько) объектов кода (байткод + обвязка);

# 4. интерпретация - Потом интерпретатор выполняет каждый объект..

6. Анализ Traceback

# 7. PEP 8 - руководство по написанию кода на Python

<https://pythonworld.ru/osnovy/pep-8-rukovodstvo-po-napisaniyu-koda-na-python.html>