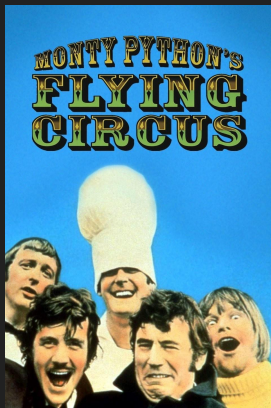









# History

- Version 1 (1994)
- Version 2 (2000)
- Version 3 (2008)



PYTHON 2.X		PYTHON 3.X
← LEGACY		FUTURE →
It is still entrenched in the software at certain companies		It will take over Python 2 by the end of 2019
 LIBRARY		LIBRARY 
Many older libraries built for Python 2 are not forwards compatible		Many of today's developers are creating libraries strictly for use with Python 3
0100 ASCII 0001		UNICODE 0000 0000 0100 0001
Strings are stored as ASCII by default		Text Strings are Unicode by default
⌊ 7/2=3		7/2=3.5 ⌊
It rounds your calculation down to the nearest whole number		This expression will result in the expected result
 print "WELCOME TO GEEKSFORGEEKS"		printl("WELCOME TO GEEKSFORGEEKS") 
It rounds your calculation down to the nearest whole number		This expression will result in the expected result



# Why Python?

- Easy to read, learn and code
- Dynamic Typing
- Free, Open Source
- Portable
- Extensive Third-Party Libraries
- Interpreted Language
- Multi paradigm
- Improved Productivity

# Disadvantages of Python

- Slow at runtime
- Runtime Errors
- Not used in the Enterprise Development
- Simplicity

[helloworld](#)

Input: QwQ

lang	code	time
<u><a href="#">c</a></u>	<u><a href="#">1.c</a></u>	1.0ms
<u><a href="#">c</a></u>	<u><a href="#">1.c</a></u>	1.3ms
<u><a href="#">c</a></u>	<u><a href="#">1.c</a></u>	1.4ms
<u><a href="#">python</a></u>	<u><a href="#">1.py</a></u>	12ms
<u><a href="#">python</a></u>	<u><a href="#">1.py</a></u>	14ms
<u><a href="#">python</a></u>	<u><a href="#">1.py</a></u>	28ms

# The Zen of Python

```
1 import this
```



<https://cs50.harvard.edu/python/2022/>  
<https://cs50.edx.org/python>



## Problem Set 3

### What to Do

1. Log into [cs50.dev](#), which is a cloud-based version of the course pre-installed. No need to download anything.
2. Execute `update50` in your codespace's terminal.
3. Submit all of the problems below:
  - [Fuel Gauge](#)
  - [Felipe's Taqueria](#)
  - [Grocery List](#)
  - [Outdated](#)

■ [Fuel Gauge](#)

- [Felipe's Taqueria](#)
- [Grocery List](#)
- [Outdated](#)

■ [Fuel Gauge](#)

■ [Felipe's Taqueria](#)

■ [Grocery List](#)

■ [Outdated](#)

### Lecture 3

- [Exceptions](#)
- [Runtime Errors](#)
- `try`
- `else`
- [Creating a Function to Get an Integer](#)
- `pass`
- [Summing Up](#)

- [Fuel Gauge](#)
- [Felipe's Taqueria](#)
- [Grocery List](#)
- [Outdated](#)

# Contact me

Telegram: @Neutrallin

Email: [neutralobs@gmail.com](mailto:neutralobs@gmail.com)

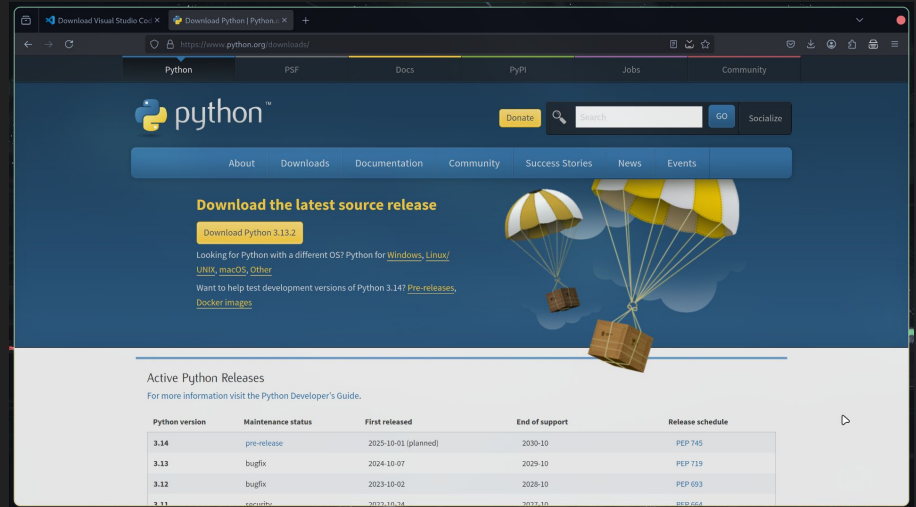
<https://github.com/Neutrallin4/PythonClass>



<https://code.visualstudio.com/download>



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



# Clean Code in Python

- Easy to understand
- More efficient
- Easier to maintain, scale, debug, and refactor

<https://testdriven.io/blog/clean-code-python/>

<https://peps.python.org/pep-0008/>  
Proposal)

PEP 8 (Python Enhancement

# Documentation

<https://docs.python.org/3/library/>

```
print(*objects, sep=' ', end='\n', file=None, flush=False)
```

Print *objects* to the text stream *file*, separated by *sep* and followed by *end*. *sep*, *end*, *file*, and *flush*, if present, must be given as keyword arguments.

All non-keyword arguments are converted to strings like [str\(\)](#) does and written to the stream, separated by *sep* and followed by *end*. Both *sep* and *end* must be strings; they can also be `None`, which means to use the default values. If no *objects* are given, [print\(\)](#) will just write *end*.

The *file* argument must be an object with a `write(string)` method; if it is not present or `None`, [sys.stdout](#) will be used. Since printed arguments are converted to text strings, [print\(\)](#) cannot be used with binary mode file objects. For these, use `file.write(...)` instead.

Output buffering is usually determined by *file*. However, if *flush* is true, the stream is forcibly flushed.

**Changed in version 3.3:** Added the *flush* keyword argument.