# PYTHON FUNDAMENTALS INTRODUCTION

by Liubov Koliasa

softserve

# More information about Python

**http://python.org/**

- documentation, tutorials, beginners guide, core distribution, ...

Wikipedia – Python

Books include:

*Learning Python* by Mark Lutz

*Python Essential Reference* by David Beazley

*Python Cookbook*, ed. by Martelli, Ravenscroft and Ascher

online at

https://python.swaroopch.com/ (Byte of Python)

https://www.coursera.org/specializations/python

https://www.udemy.com/python-101-beginners-coding-bootcamp-free-course/

https://www.w3schools.com/python/

softserve

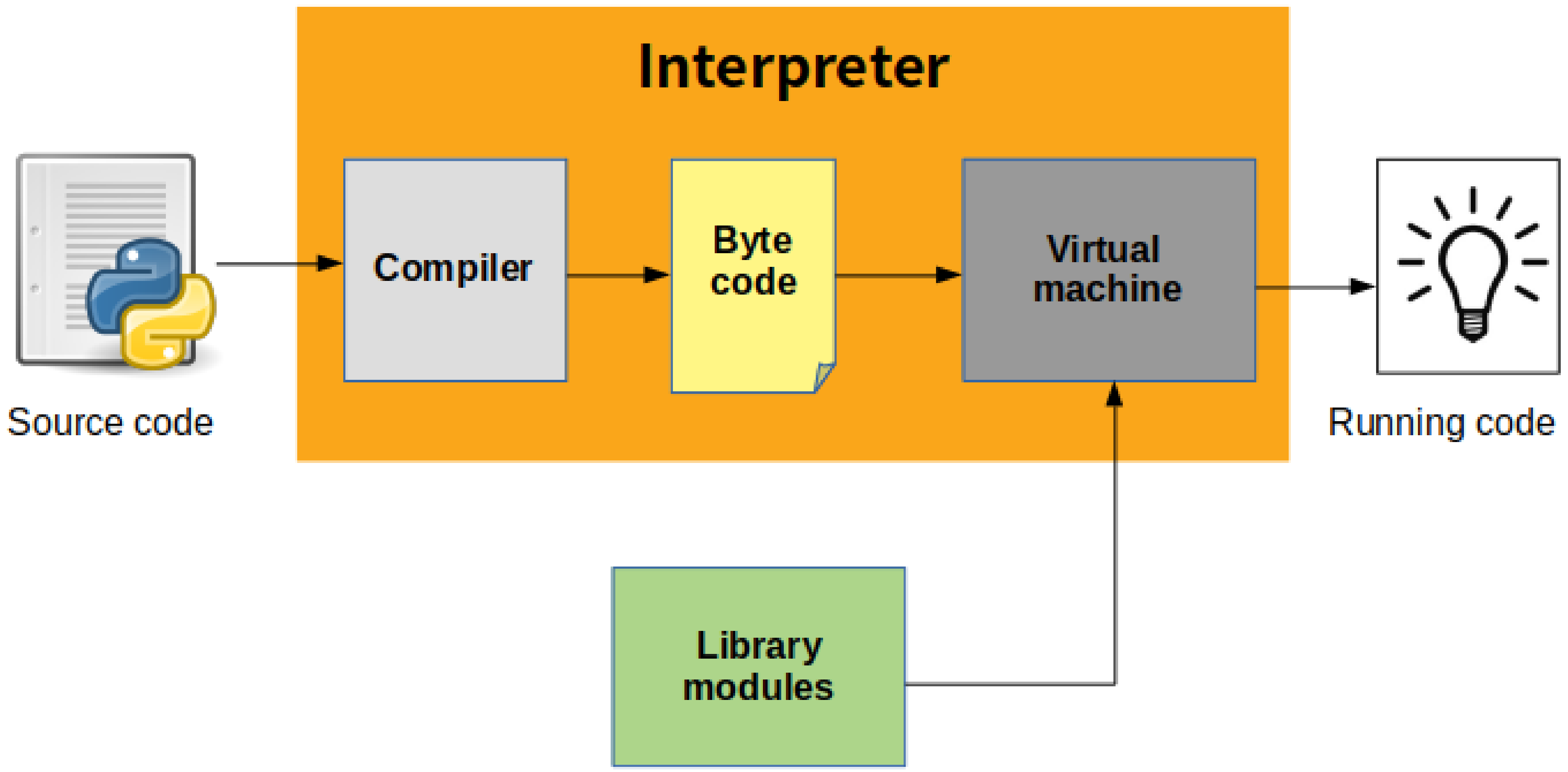
## Python features

- **Python** is an **interpreted**, **interactive**, **object-oriented** programming language.
- **Python** is a **general-purpose**, **high-level** programming language whose design philosophy emphasizes code readability.
- **Python** aims to combine "remarkable power with very clear syntax", and its standard library is large and comprehensive.
- Its use of indentation for block delimiters is unusual among popular programming languages.

- **Python** features a **dynamic type** system and automatic **memory management**. It supports multiple **programming paradigms**, including **object oriented**, **imperative**, **functional** and **procedural**, and has a large and comprehensive **standard library**.

softserve

# Python Interpreter



serve

## Simplicity and conciseness

Python:

```
file = open('file.txt')
content = file.read()
```

Another programming language:

```
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Paths;
public class Main {
    public static void main(String[] args) throws IOException {
        String content = new
String(Files.readAllBytes(Paths.get("file.txt")));
    }
}
```

softserve

## Python history

**Python** created by Guido van Rossum and first released in 1991.

Python 1.0 - January 1994
- Python 1.5 - December 31, 1997
- Python 1.6 - September 5, 2000

**Python 2.0** - October 16, 2000
- Python 2.1 - April 17, 2001
- Python 2.2 - December 21, 2001
- Python 2.3 - July 29, 2003
- Python 2.4 - November 30, 2004
- Python 2.5 - September 19, 2006
- Python 2.6 - October 1, 2008
- Python 2.7 - July 3, 2010

**Python 3.0** - December 3, 2008
- Python 3.1 - June 27, 2009
- Python 3.2 - February 20, 2011
- Python 3.3 - September 29, 2012
- Python 3.4 - March 16, 2014
- Python 3.5 - September 13, 2015
- Python 3.6 - December 23, 2016
- Python 3.7 - June 27, 2018
- Python 3.8 - October 14, 2019
- Python 3.9 - October 5, 2020
- Python 3.10 - October 4, 2021
- Python 3.11 - October 24, 2022

softserve

## Why Python ...?

softserve

# Python Overview

**Cisco, Inc –** Web security (e-mail, www)

**Yahoo** (Maps, Groups)

**Google** – many components of the Google spider and search engine are written in Python

**YouTube** – entirely written in Python

**Zope Corporation** – has developed a powerful Web application framework server using Python

**Linux Weekly News, Instagram**

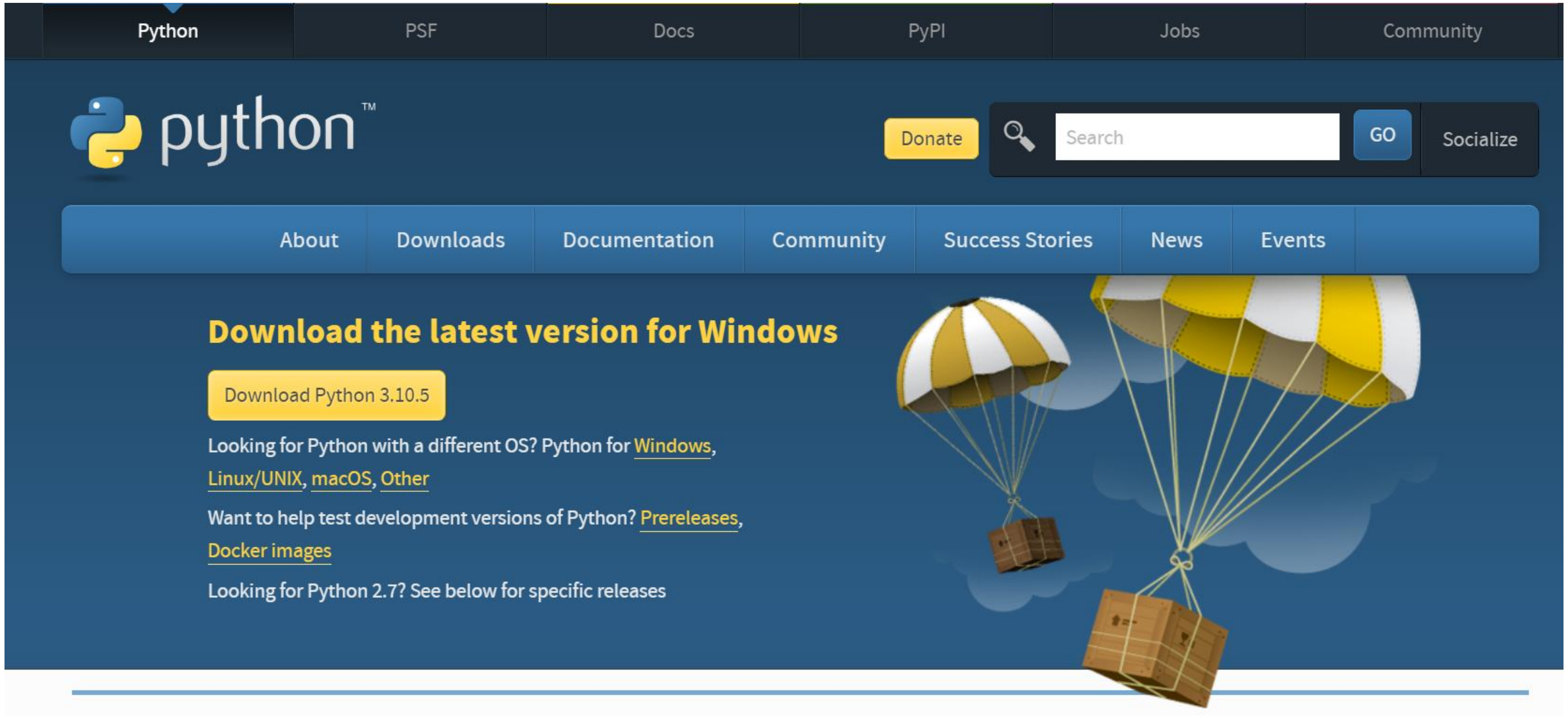
George Lucas' Film Company - **Industry Light and Magic** – uses Python in the production of their FX

**Walt Disney Feature Animation, Pinterest**

**NASA –** uses Python in its integrated Planning System and Mission Control Center. Python is going to replace other tools written in Perl and shell dialects

# Python download

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



Download the latest version for Windows

Download Python 3.10.5

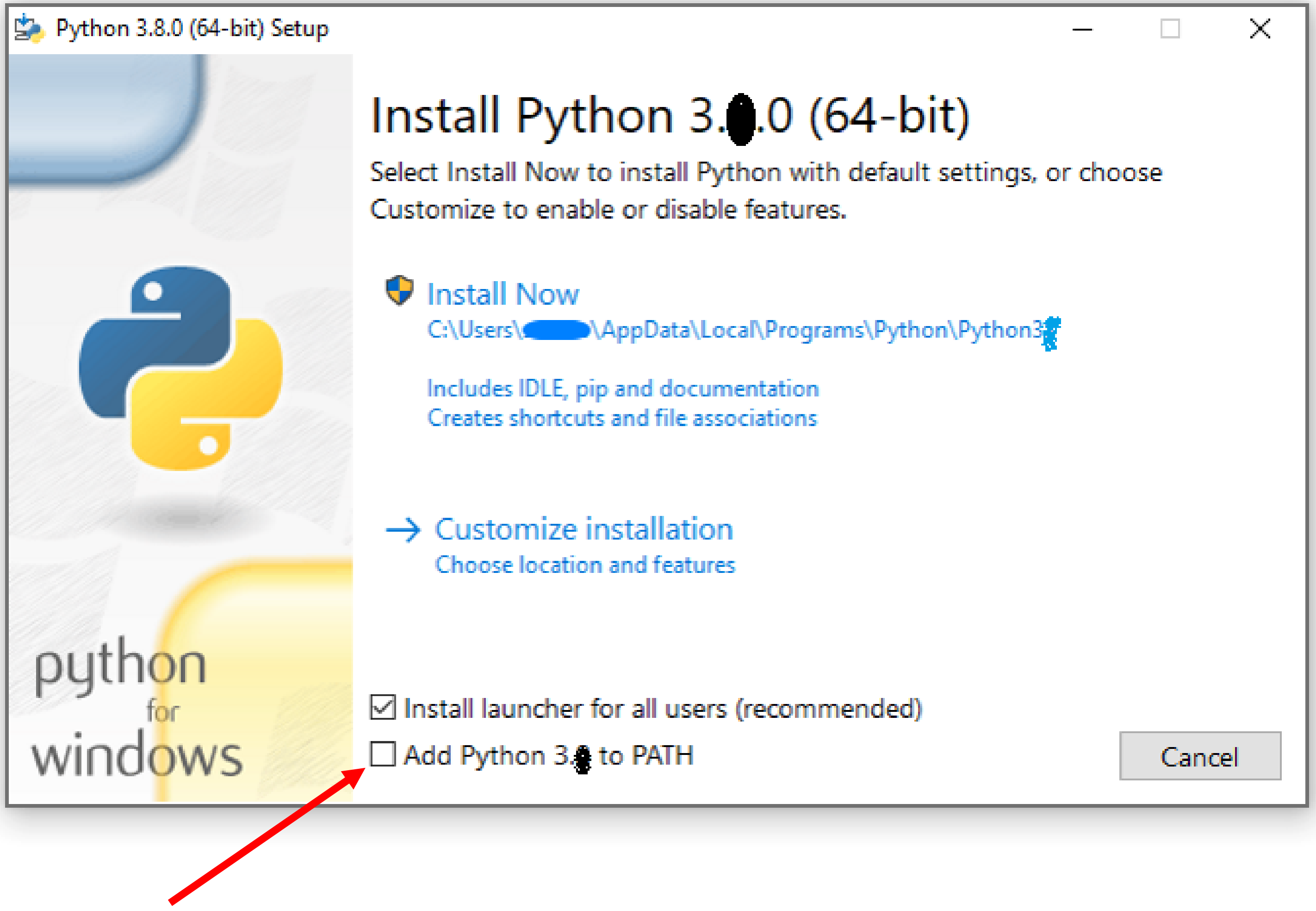
Looking for Python with a different OS? Python for Windows, Linux/UNIX, macOS, Other

Want to help test development versions of Python? Prereleases, Docker images

Looking for Python 2.7? See below for specific releases

softserve

# Python download



Python 3.8.0 (64-bit) Setup

Install Python 3.[redacted].0 (64-bit)

Select Install Now to install Python with default settings, or choose Customize to enable or disable features.

Install Now
C:\Users\[redacted]\AppData\Local\Programs\Python\Python3[redacted]

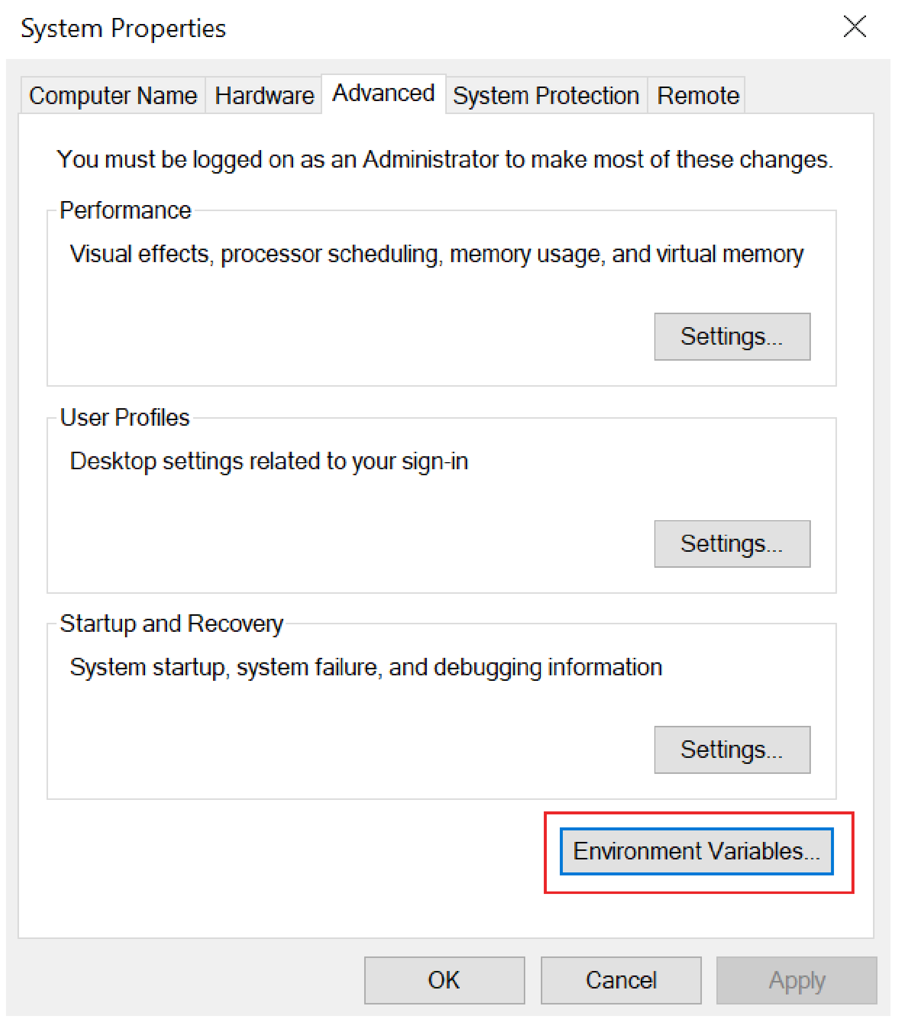
Includes IDLE, pip and documentation
Creates shortcuts and file associations

→ Customize installation
Choose location and features

☑ Install launcher for all users (recommended)
☐ Add Python 3.[redacted] to PATH

softserve

# Install Python



System Properties

Computer Name | Hardware | Advanced | System Protection | Remote

You must be logged on as an Administrator to make most of these changes.

Performance

Visual effects, processor scheduling, memory usage, and virtual memory

Settings...

User Profiles

Desktop settings related to your sign-in

Settings...

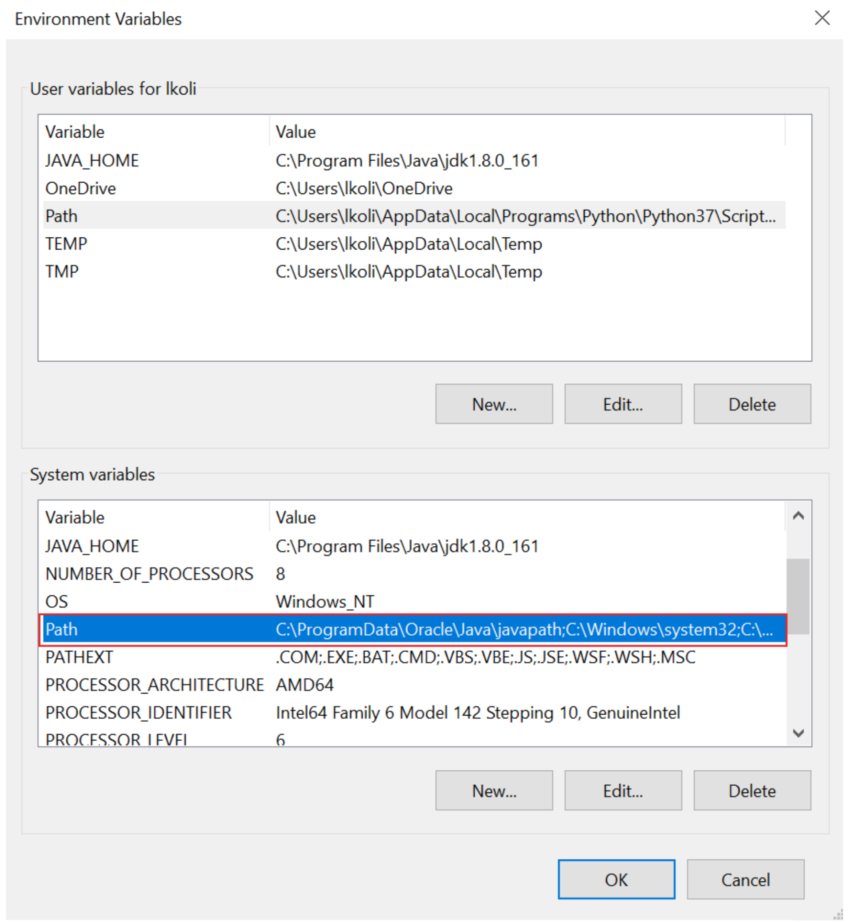
Startup and Recovery

System startup, system failure, and debugging information

Settings...

Environment Variables...

OK | Cancel | Apply



Environment Variables

User variables for lkoli

| Variable | Value |
|---|---|
| JAVA_HOME | C:\Program Files\Java\jdk1.8.0_161 |
| OneDrive | C:\Users\lkoli\OneDrive |
| Path | C:\Users\lkoli\AppData\Local\Programs\Python\Python37\Script... |
| TEMP | C:\Users\lkoli\AppData\Local\Temp |
| TMP | C:\Users\lkoli\AppData\Local\Temp |

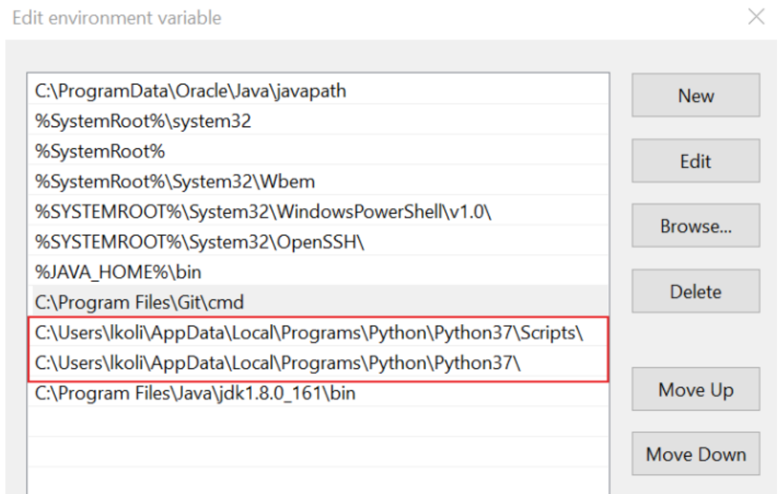
New... | Edit... | Delete

System variables

| Variable | Value |
|---|---|
| JAVA_HOME | C:\Program Files\Java\jdk1.8.0_161 |
| NUMBER_OF_PROCESSORS | 8 |
| OS | Windows_NT |
| Path | C:\ProgramData\Oracle\Java\javapath;C:\Windows\system32;C:\... |
| PATHEXT | .COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC |
| PROCESSOR_ARCHITECTURE | AMD64 |
| PROCESSOR_IDENTIFIER | Intel64 Family 6 Model 142 Stepping 10, GenuineIntel |
| PROCESSOR_LEVEL | 6 |

New... | Edit... | Delete

OK | Cancel



Edit environment variable

```
C:\ProgramData\Oracle\Java\javapath
%SystemRoot%\system32
%SystemRoot%
%SystemRoot%\System32\Wbem
%SYSTEMROOT%\System32\WindowsPowerShell\v1.0\
%SYSTEMROOT%\System32\OpenSSH\
%JAVA_HOME%\bin
C:\Program Files\Git\cmd
C:\Users\lkoli\AppData\Local\Programs\Python\Python37\Scripts\
C:\Users\lkoli\AppData\Local\Programs\Python\Python37\
C:\Program Files\Java\jdk1.8.0_161\bin
```

Add Python to the PATH Environmental Variable

You must append your **installation path** (example: C:\Users\*User*\AppData\Local\Programs\Python\Python3*)

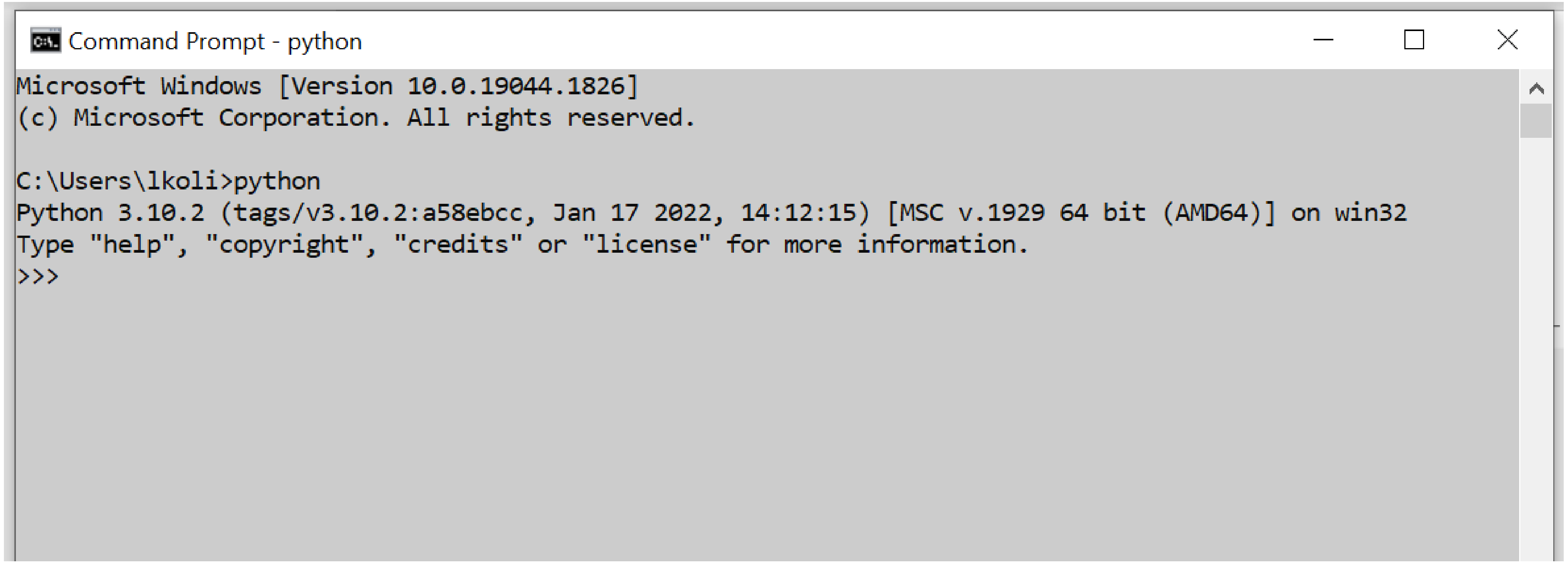
to the **PATH variable in System**

softserve

# Install Python

Next step:

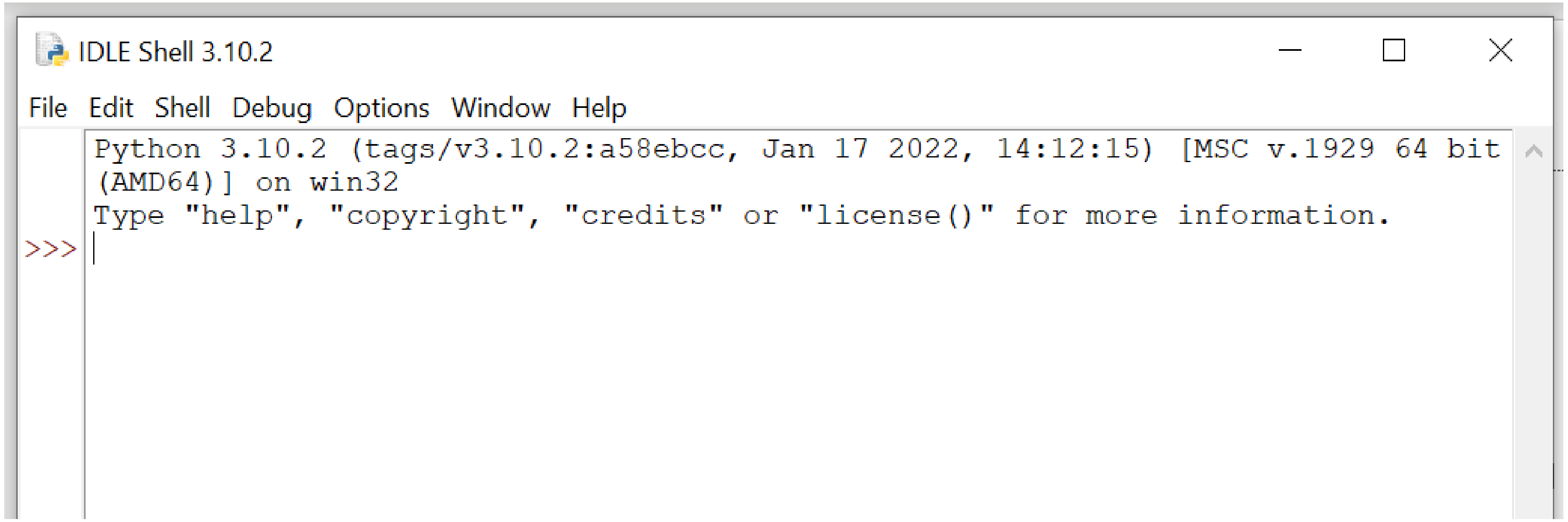
open your command line and type **`python`** or **`py`**



```
Microsoft Windows [Version 10.0.19044.1826]
(c) Microsoft Corporation. All rights reserved.

C:\Users\lkoli>python
Python 3.10.2 (tags/v3.10.2:a58ebcc, Jan 17 2022, 14:12:15) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

softserve

## Python Interactive Shell



```
Python 3.10.2 (tags/v3.10.2:a58ebcc, Jan 17 2022, 14:12:15) [MSC v.1929 64 bit
(AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> 
```

Python 3.10.2rc1 (v3.7.1rc1:2064bcf6ce, Jul 26 2022, 15:15:36) [MSC v.1914 64 bit (AMD64)] on win32

Type "help", "copyright", "credits" or "license()" for more information.

>>>

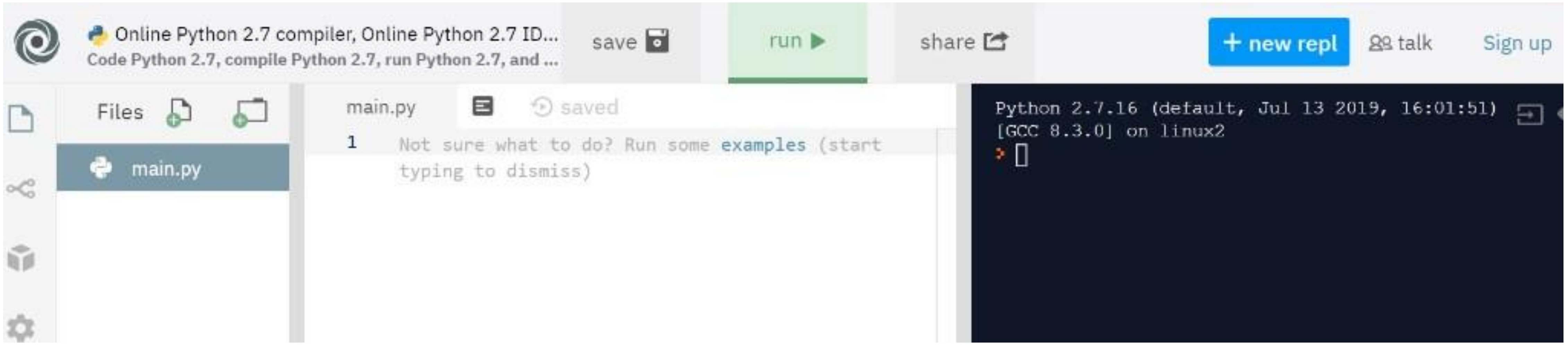
softserve

## Online Python compiler

- https://replit.com/languages/python3
- https://paiza.io/projects/sepDWD3s9TLX_8GKIvvbXA?language=python3
- https://ideone.com/
- https://www.tutorialspoint.com/execute_python_online.php
- https://www.jdoodle.com/python3-programming-online
- https://www.programiz.com/python-programming/online-compiler/
- https://www.onlinegdb.com/online_python_compiler
- https://www.online-python.com/

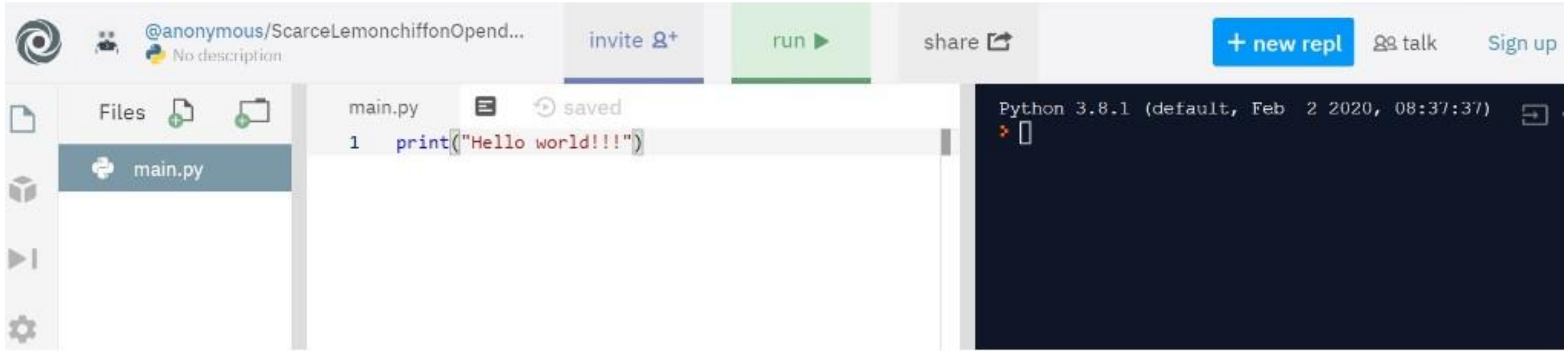
softserve

# Online Python Editor – for example Repl.it


Online Python 2.7 compiler, Online Python 2.7 ID...
Code Python 2.7, compile Python 2.7, run Python 2.7, and ...
save
run
share
+ new repl
talk
Sign up
Files
main.py
main.py
saved

```
Not sure what to do? Run some examples (start
typing to dismiss)
```

```
Python 2.7.16 (default, Jul 13 2019, 16:01:51)
[GCC 8.3.0] on linux2
```



```
print("Hello world!!!")
```

```
Python 3.8.1 (default, Feb  2 2020, 08:37:37)
```

softserve

# Environment, Development Tools

- Eclipse + PyDev
- Sublime Text
- **Atom**
- GNU Emacs
- Vi / Vim
- Visual Studio
- **Visual Studio Code**
- **PyCharm** by JetBrains
- Spyder
- Thonny

softserve

## Environment, Development Tools

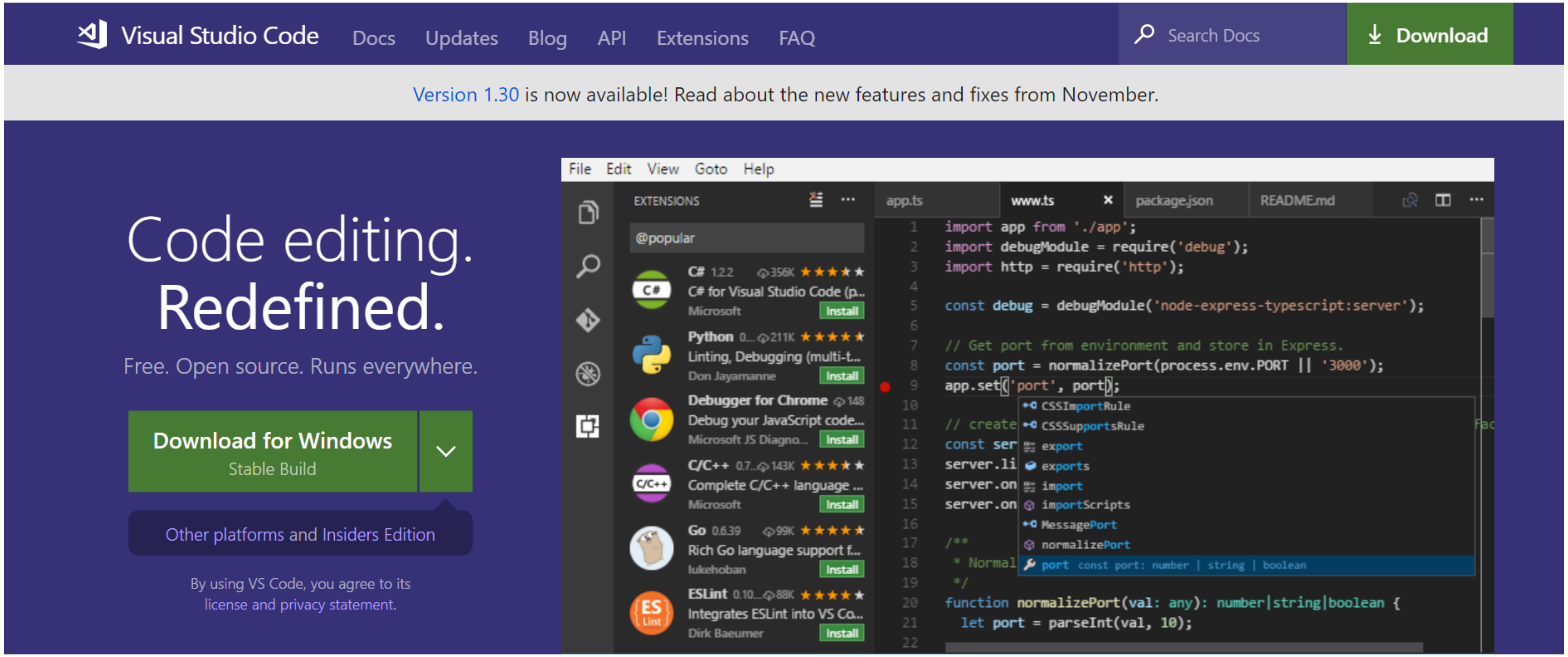
IDLE(Integrated Development and Learning Environment) multi-window colorized source browser, autoindent, autocompletion, tool tips, code context panel, search in files, class and path browsers, debugger, executes code in clean separate subprocess with one keystroke. 100% pure Python, part of Python 2.x and 3.x distributions (may be packaged separately in some situations).

PyCharm Community is a free open-source IDE with a smart Python editor providing quick code navigation, code completion, refactoring, unit testing and debugger. Commercial Professional edition fully supports Web development with Django, Flask, Mako

Visual Studio Code is an integrated development environment (IDE) from Microsoft.

# Visual Studio Code

[Visual Studio Code](#)



Visual Studio Code Docs Updates Blog API Extensions FAQ Search Docs Download

Version 1.30 is now available! Read about the new features and fixes from November.

Code editing.
Redefined.

Free. Open source. Runs everywhere.

Download for Windows
Stable Build

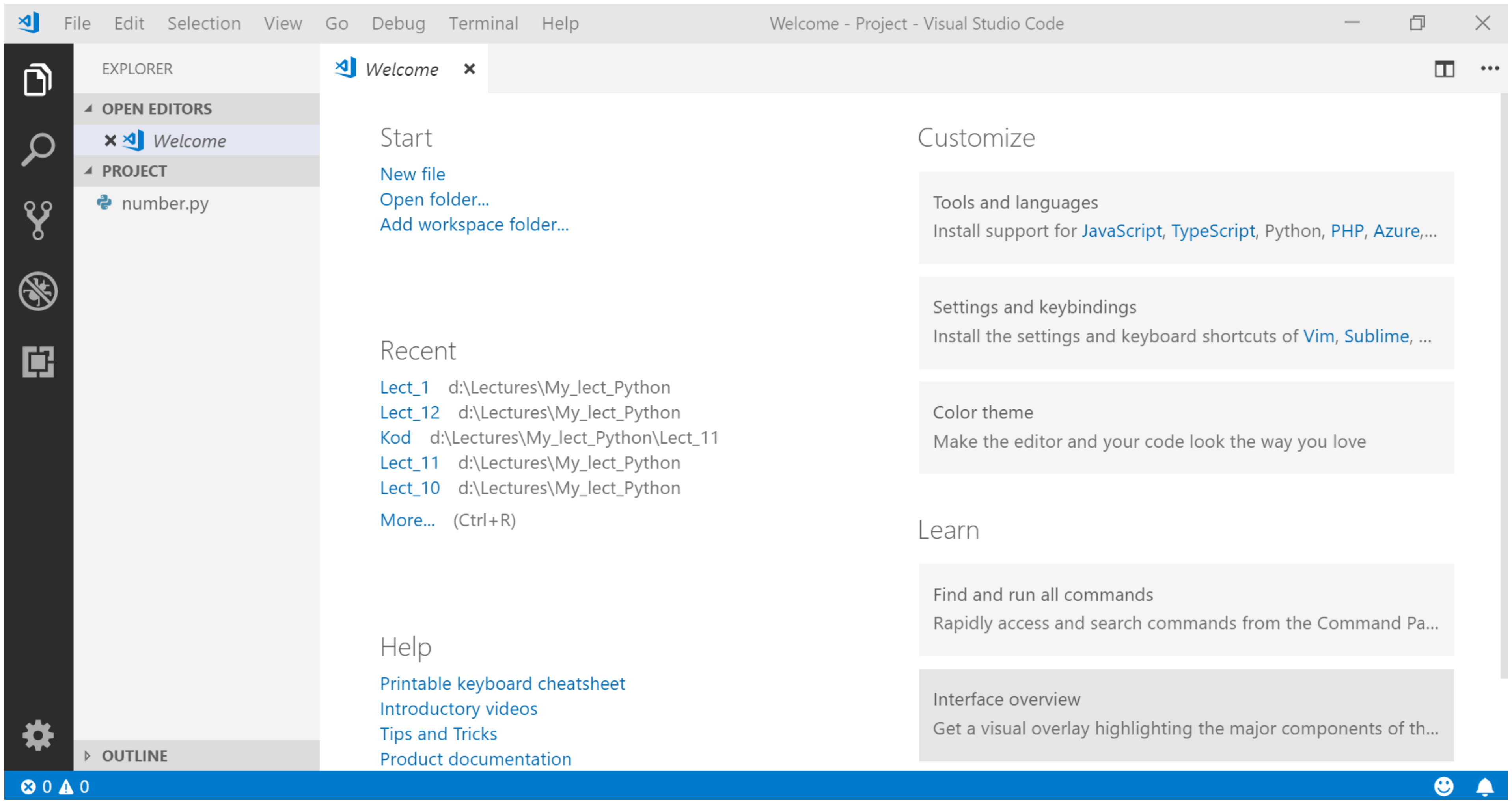
Other platforms and Insiders Edition

By using VS Code, you agree to its license and privacy statement.

```
import app from './app';
import debugModule = require('debug');
import http = require('http');

const debug = debugModule('node-express-typescript:server');

// Get port from environment and store in Express.
const port = normalizePort(process.env.PORT || '3000');
app.set('port', port);

// create
const ser
server.li
server.on
server.on

/**
 * Normal
 */
function normalizePort(val: any): number|string|boolean {
  let port = parseInt(val, 10);
```

softserve

# Visual Studio Code



Start

New file
Open folder...
Add workspace folder...

Recent

| | |
|---|---|
| Lect_1 | d:\Lectures\My_lect_Python |
| Lect_12 | d:\Lectures\My_lect_Python |
| Kod | d:\Lectures\My_lect_Python\Lect_11 |
| Lect_11 | d:\Lectures\My_lect_Python |
| Lect_10 | d:\Lectures\My_lect_Python |

More... (Ctrl+R)

Help

Printable keyboard cheatsheet
Introductory videos
Tips and Tricks
Product documentation

Customize

Tools and languages
Install support for JavaScript, TypeScript, Python, PHP, Azure,...

Settings and keybindings
Install the settings and keyboard shortcuts of Vim, Sublime, ...

Color theme
Make the editor and your code look the way you love

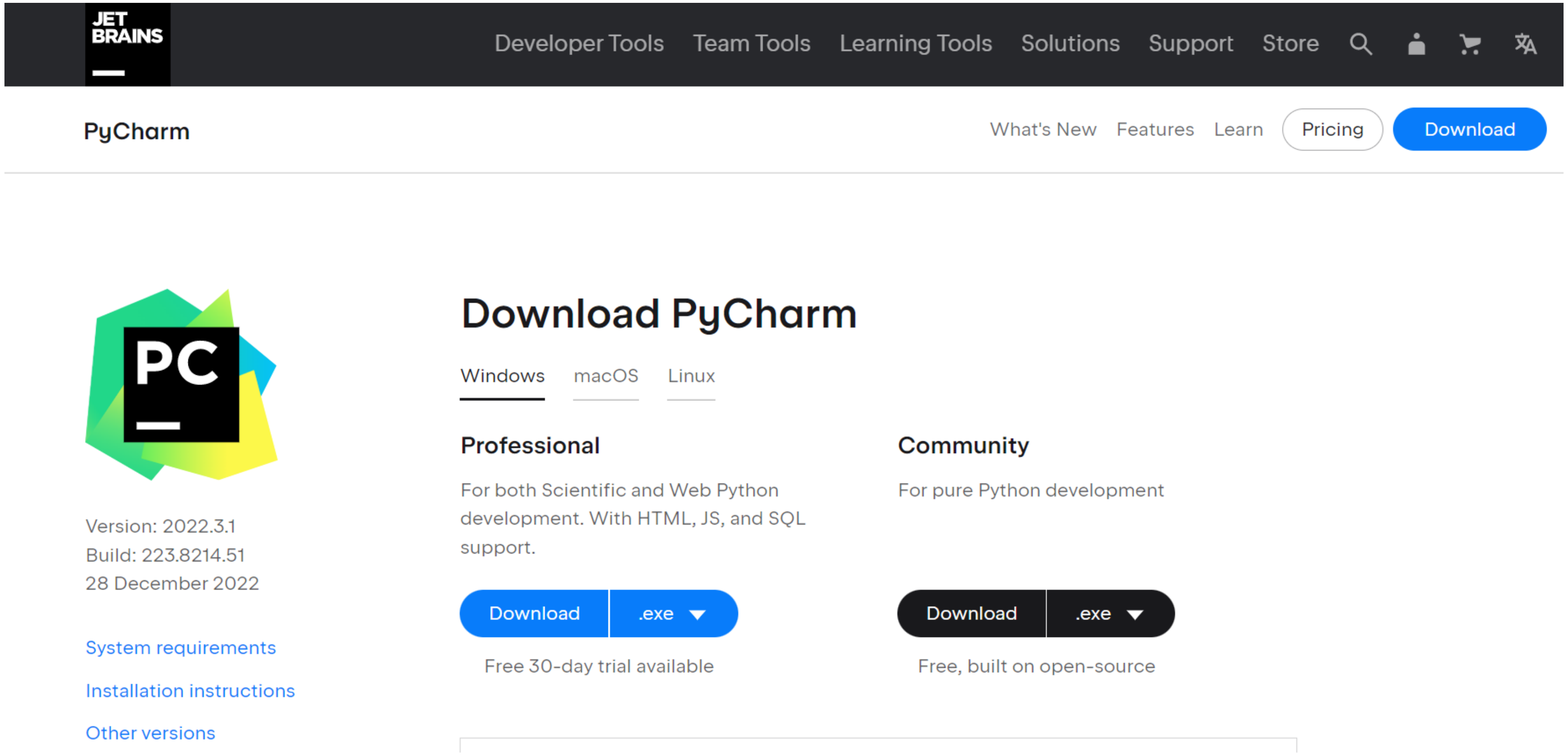
Learn

Find and run all commands
Rapidly access and search commands from the Command Pa...

Interface overview
Get a visual overlay highlighting the major components of th...

# PyCharm

PyCharm



## Download PyCharm

**Professional**

For both Scientific and Web Python development. With HTML, JS, and SQL support.

Free 30-day trial available

**Community**

For pure Python development

Free, built on open-source

Version: 2022.3.1
Build: 223.8214.51
28 December 2022

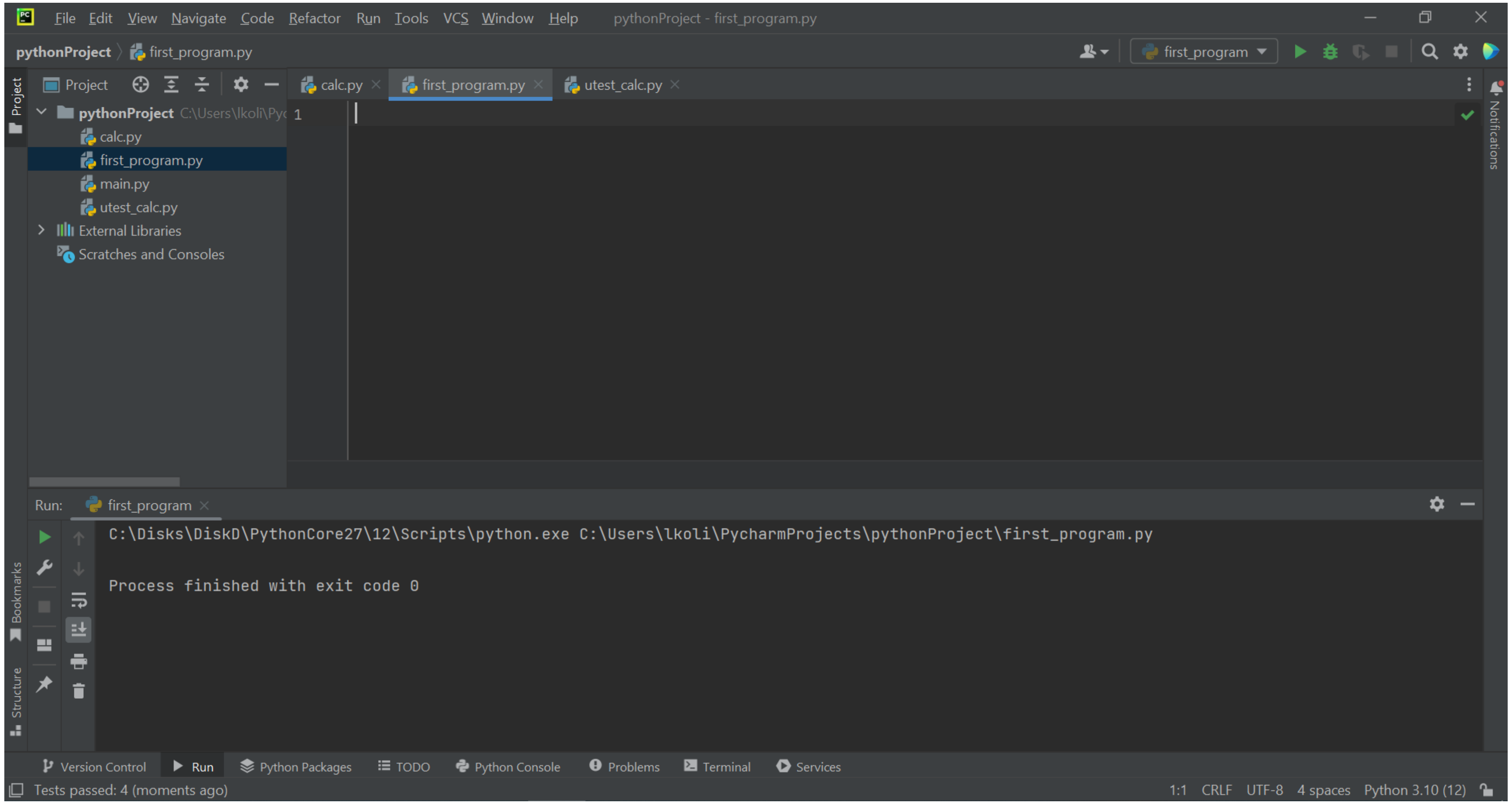
System requirements

Installation instructions

Other versions

softserve

# PyCharm



```
C:\Disks\DiskD\PythonCore27\12\Scripts\python.exe C:\Users\lkoli\PycharmProjects\pythonProject\first_program.py

Process finished with exit code 0
```

softserve

# Python philosophy: The Zen of Python, by Tim Peters

**>>> import this**

- 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!

softserve

# Python syntax

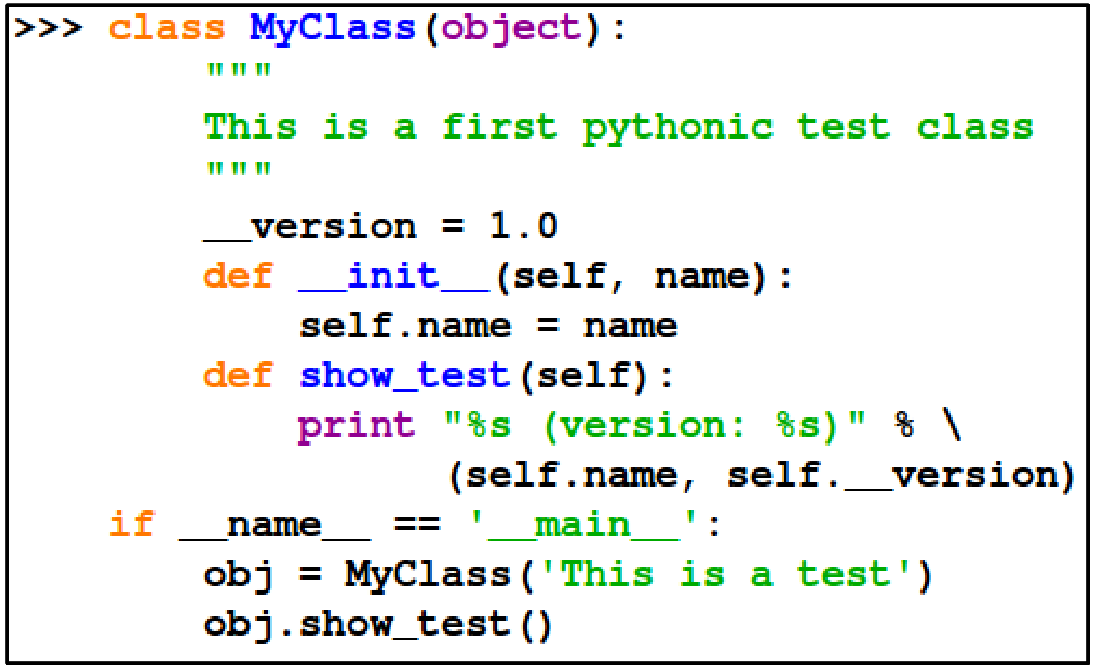
softserve

# Python Syntax

Python is an easy to learn, powerful programming language. It has efficient high-level data structures and a simple but effective approach to object-oriented programming. Python's elegant syntax and dynamic typing, together with its interpreted nature, make it an ideal language for scripting and rapid application development in many areas on most platforms.

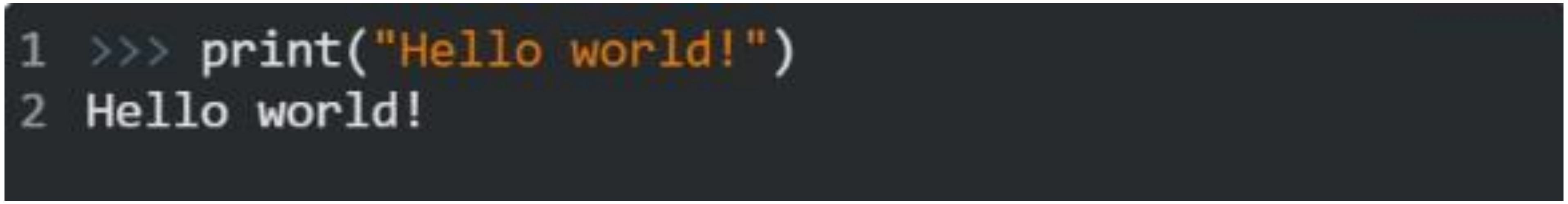
**Indentation** - Python uses whitespace indentation, rather than curly braces or keywords, to delimit blocks (a feature also known as the off-side rule):

- increase in indentation comes after certain statements;
- decrease in indentation signifies the end of the current block
- Use '\' when must go to next line prematurely



```
>>> class MyClass(object):
        """
        This is a first pythonic test class
        """
        __version = 1.0
        def __init__(self, name):
            self.name = name
        def show_test(self):
            print "%s (version: %s)" % \
                  (self.name, self.__version)
    if __name__ == '__main__':
        obj = MyClass('This is a test')
        obj.show_test()
```

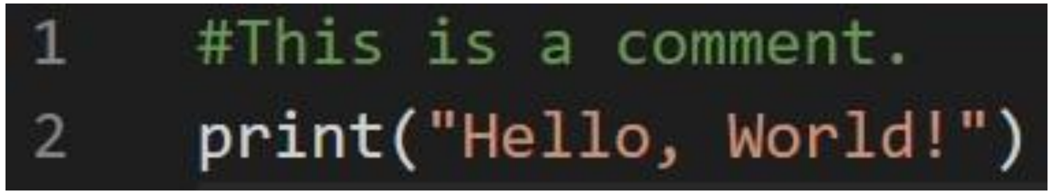
## Getting Started With Python



```
>>> print("Hello world!")
Hello world!
```

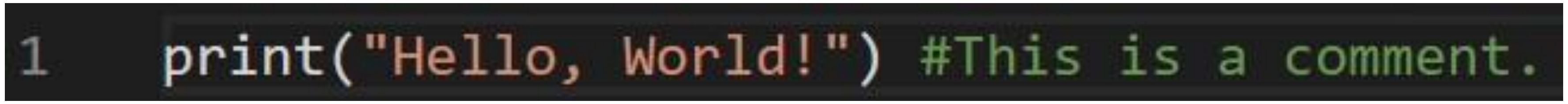
# Python Comments

Comments starts with #, and Python will ignore them:



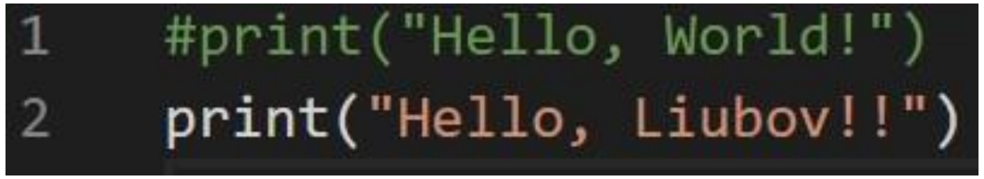
```python
#This is a comment.
print("Hello, World!")
```

Comments can be placed at the end of a line, and Python will ignore the rest of the line:



```python
print("Hello, World!") #This is a comment.
```

Comments does not have to be text to explain the code, it can also be used to prevent Python from executing code:

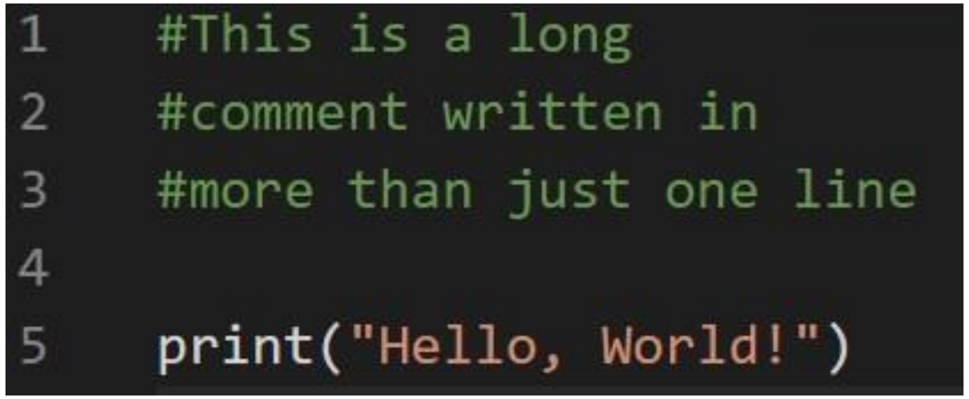


```python
#print("Hello, World!")
print("Hello, Liubov!!")
```

softserve

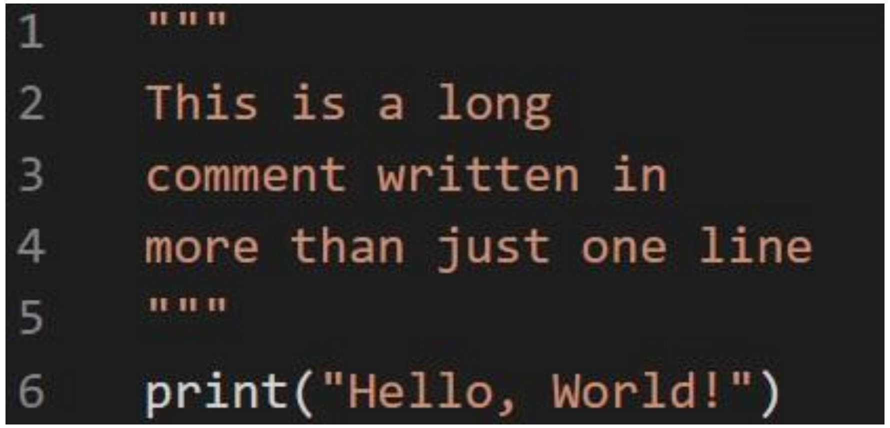
## Multi Line Comments

Python does not really have a syntax for **multi line comments**. To add a multiline comment you could insert a **#** for each line:



```python
#This is a long
#comment written in
#more than just one line

print("Hello, World!")
```

**Another way** is to use triple quotes, either **'''** or **"""**.
Since Python will ignore string literals that are not assigned to a variable, you can add a multiline string (triple quotes) in your code, and place your comment inside it:



```python
"""
This is a long
comment written in
more than just one line
"""
print("Hello, World!")
```

## Python Variables

In most of the programming languages a **variable** is a named location used to store data in the memory. Each variable must have a unique name called identifier. It is helpful to think of variables as container that hold data which can be changed later throughout programming.

In Python we don't assign values to the variables, where as Python gives the **reference** of the object (value) to the variable.

In Python, variables **do not need declaration** to reserve memory space. The **"variable declaration"** or **"variable initialization"** happens **automatically** when we assign a value to a variable.

A process in which a variable is set to its first value is called ***initialization***.

softserve

# Python Keywords

Keywords are the reserved words in Python.

In Python, keywords are case sensitive.

There are 35 keywords:

```
>>> help("keywords")
```

or

```
>>> import keyword
>>> keyword.kwlist
```

| True | False | None | async | await |
|---|---|---|---|---|
| and | def | for | is | raise |
| as | del | from | lambda | return |
| assert | elif | global | nonlocal | try |
| break | else | if | not | while |
| class | except | import | or | with |
| continue | finally | in | pass | yield |

softserve

# Python Identifiers

Identifier is the name given to entities like class, functions, variables etc. in Python. It helps differentiating one entity from another.

**Rules for writing identifiers**

- Identifiers can be a combination of letters in lowercase (**a to z**) or uppercase (**A to Z**) or digits (**0 to 9**) or an underscore (_). Names like **myClass**, **var_1** and **print_this_to_screen**, all are valid example.
- An identifier **cannot start with a digit**. **1variable is invalid**, but **variable1 is perfectly fine**.
- Keywords cannot be used as identifiers.
- We cannot use special symbols like **!, @, #, $, %** etc. in our identifier.
- Identifier can be of any length.

# Python Identifiers

**Things to care about**

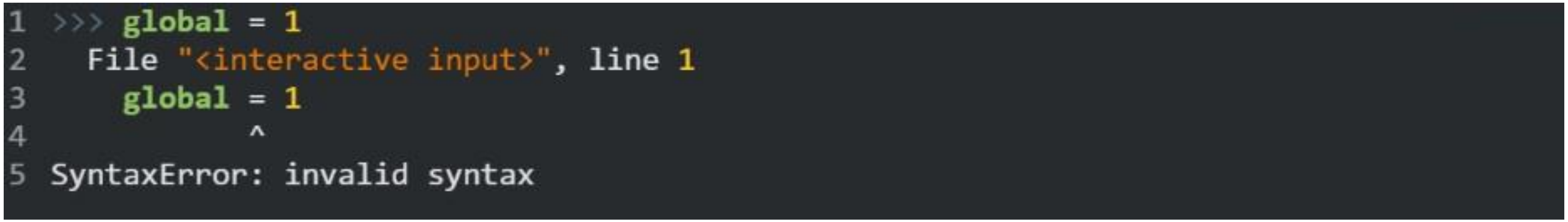
Python is a **case-sensitive language**. This means, **Variable** and **variable** are **not the same**. Always name identifiers that make sense.

While, **c = 10** is valid. Writing **count = 10** would make more sense and it would be easier to figure out what it does even when you look at your code after a long gap.

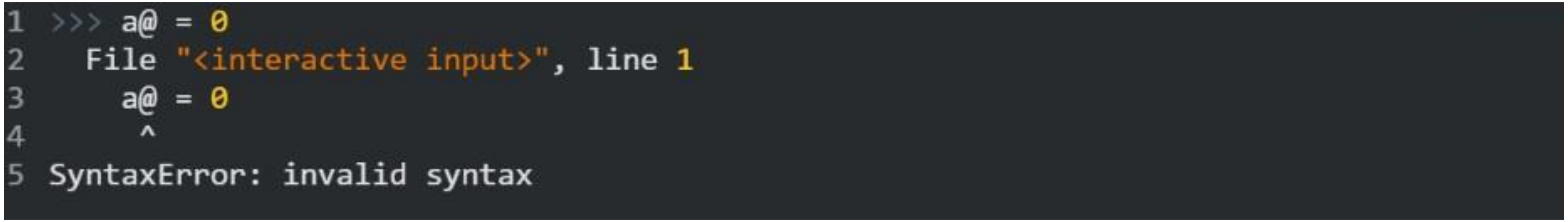
Multiple words can be separated using an underscore, **this_is_a_long_variable** (**snake case**)

We can also use **camel-case** style of writing, i.e., capitalize every first letter of the word except the initial word without any spaces. For example: **camelCaseExample**

# Python Keywords and Identifiers



```
>>> global = 1
  File "<interactive input>", line 1
    global = 1
           ^
SyntaxError: invalid syntax
```



```
>>> a@ = 0
  File "<interactive input>", line 1
    a@ = 0
     ^
SyntaxError: invalid syntax
```

softserve

# Python Syntax - Enough to Understand the Code

- Assignment uses **=** and comparison uses **==**.
- For numbers **+ - * / % //** are as expected.
- Special use of **+** for string concatenation.
- Special use of **%** for string formatting.
- Logical operators are words (**and, or, not**) not symbols (**&&, ||, !** ).
- First assignment to a variable will create it.
- Variable types don't need to be declared.
- Python figures out the variable types on its own.

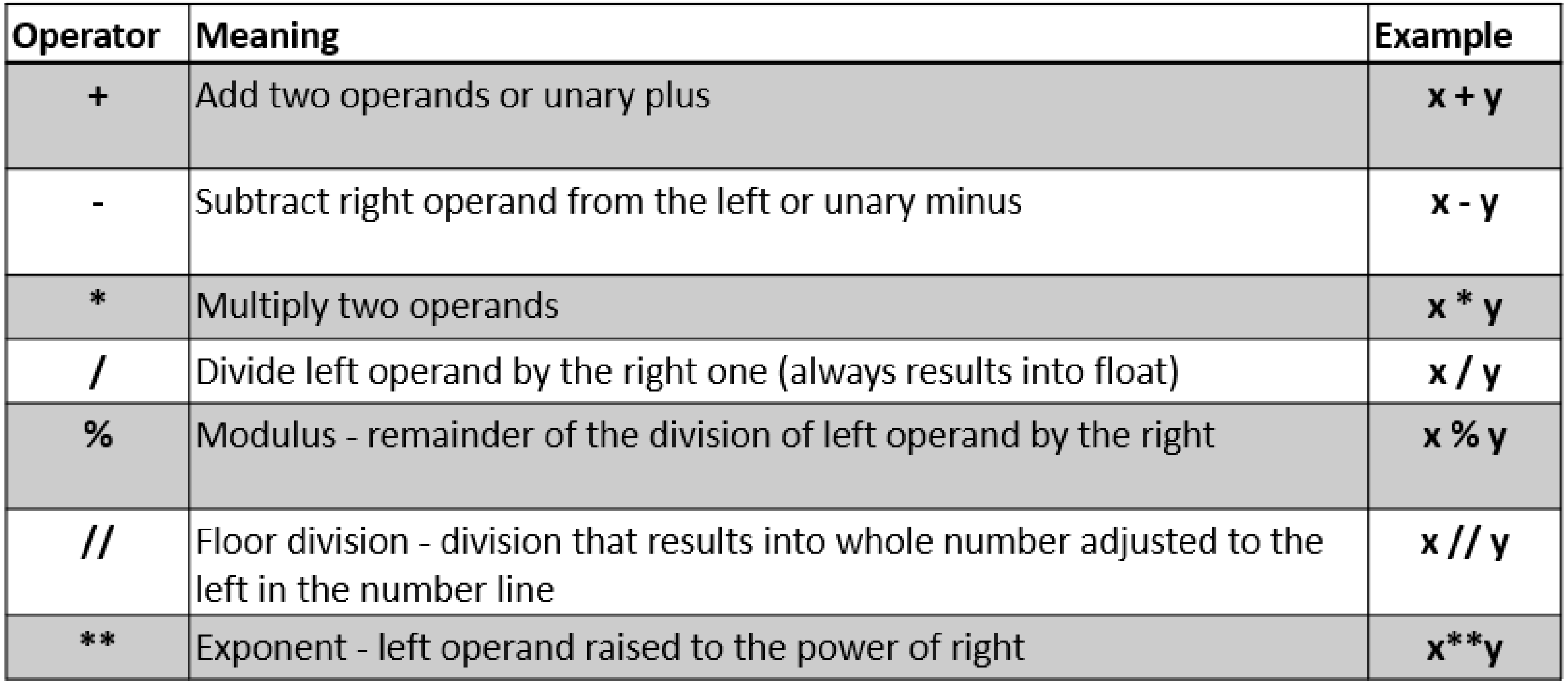
**Variables**

- No need to declare
- The variable name is case sensitive: 'val' is not the same as 'Val'
- Variables are created when they are assigned
- A variable can be reassigned to whatever, whenever (functions, modules, classes)
- The type of the variable is determined by Python

softserve

## Arithmetic operators

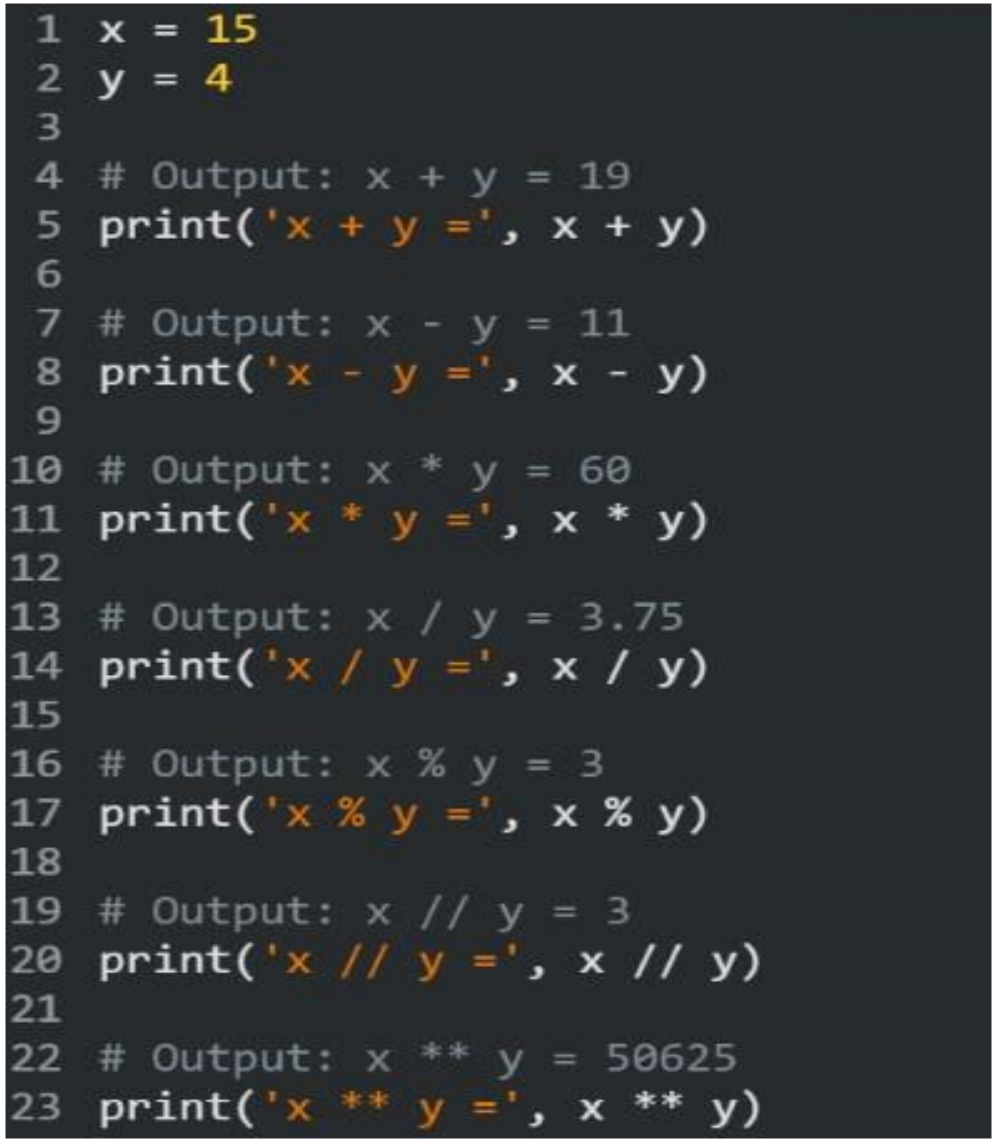
Arithmetic operators are used to perform mathematical operations like addition, subtraction, multiplication etc.



| Operator | Meaning | Example |
|---|---|---|
| + | Add two operands or unary plus | x + y |
| - | Subtract right operand from the left or unary minus | x - y |
| * | Multiply two operands | x * y |
| / | Divide left operand by the right one (always results into float) | x / y |
| % | Modulus - remainder of the division of left operand by the right | x % y |
| // | Floor division - division that results into whole number adjusted to the left in the number line | x // y |
| ** | Exponent - left operand raised to the power of right | x**y |

softserve

# Arithmetic operators



```python
x = 15
y = 4

# Output: x + y = 19
print('x + y =', x + y)

# Output: x - y = 11
print('x - y =', x - y)

# Output: x * y = 60
print('x * y =', x * y)

# Output: x / y = 3.75
print('x / y =', x / y)

# Output: x % y = 3
print('x % y =', x % y)

# Output: x // y = 3
print('x // y =', x // y)

# Output: x ** y = 50625
print('x ** y =', x ** y)
```

## Arithmetic operators

```
number = 3 + 4 * 5 ** 2 + 7
print(number)



number = (3 + 4) * (5 ** 2 + 7)
print(number)



number = 2**3**2
print(number)



number = (2**3)**2
print(number)
```

softserve

# PEP8

softserve

## Python Enhancement Proposal

- **PEP 20** – The Zen of Python
- **PEP 8** – Style Guide for Python Code: this PEP defines the coding convention when writing Python.
- **PEP 257** – Docstring Conventions: conventions for Docstring in Python

**Code is read more often than it is written**. Code should always be written in a way that promotes readability.

[https://peps.python.org/](https://peps.python.org/)

softserve

# Indentation and Spaces

- **Use 4 spaces per indentation level**
- **Align with opening delimiter**
- **Still, readability counts more than sticking to the rules**

```python
foo = long_function_name(var_one, var_two,
                         var_three, var_four)
```

```python
my_list = [
    1, 2, 3,
    4, 5, 6,
]
result = some_function_that_takes_arguments(
    'a', 'b', 'c',
    'd', 'e', 'f',
)
```

softserve

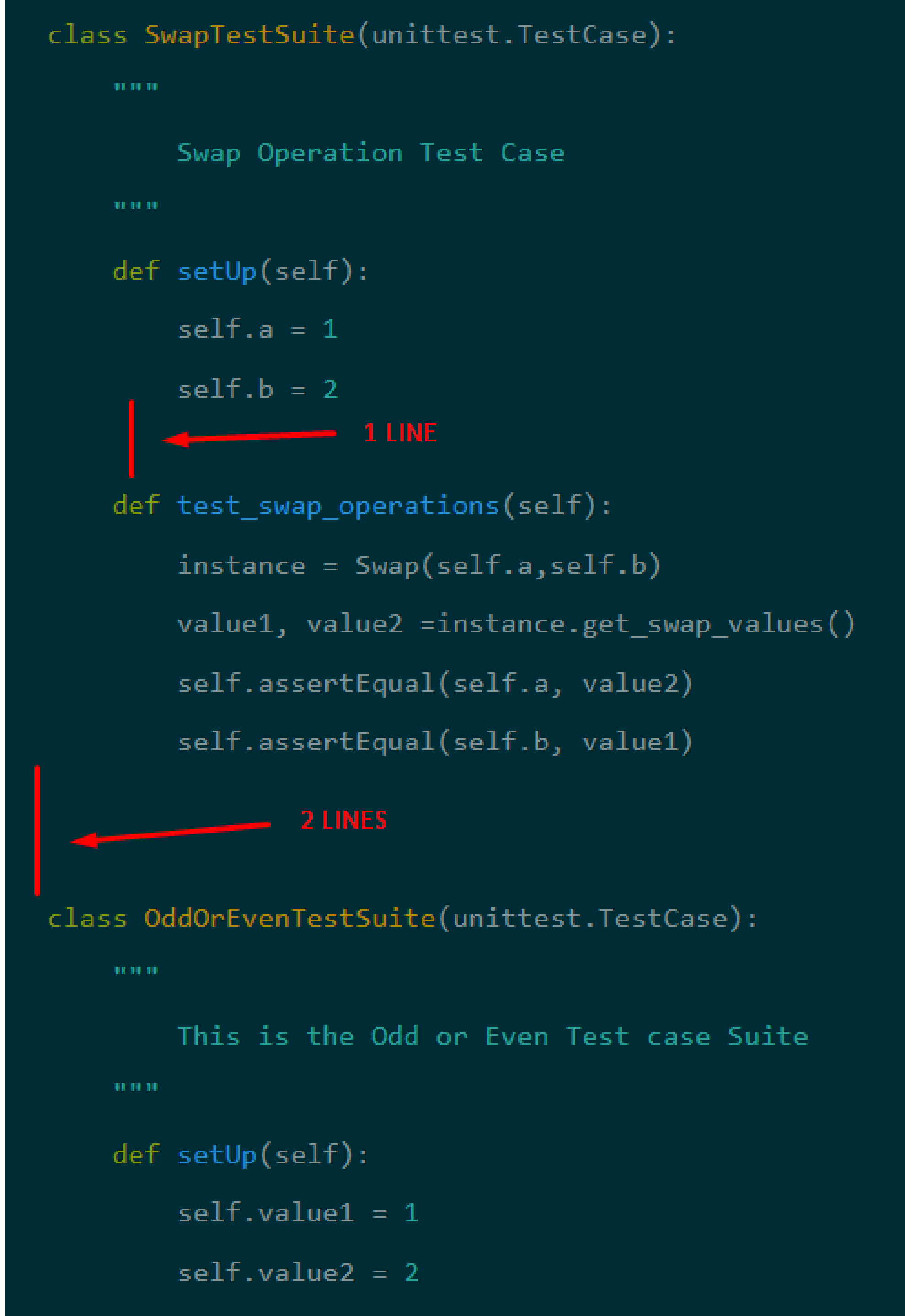
## Maximum Line Length

- **Limit all lines to a maximum of 79 characters.**
- **For flowing long blocks of text (docstrings or comments), the line length should be limited to 72 characters.**
- **This helps when reading Python code and keeping it in editor, review, and debugging tools.**

softserve

## Blank Lines

- **Two blank lines should be both before and after class definitions.**
- **One blank line should be both before and after method definitions.**
- **You should use blank lines conservatively within your code to separate groups of functions.**



```
class SwapTestSuite(unittest.TestCase):
    """
        Swap Operation Test Case
    """
    def setUp(self):
        self.a = 1
        self.b = 2

    def test_swap_operations(self):
        instance = Swap(self.a,self.b)
        value1, value2 =instance.get_swap_values()
        self.assertEqual(self.a, value2)
        self.assertEqual(self.b, value1)


class OddOrEvenTestSuite(unittest.TestCase):
    """
        This is the Odd or Even Test case Suite
    """
    def setUp(self):
        self.value1 = 1
        self.value2 = 2
```

## Source File Encoding

- Code in the core Python distribution should always use UTF-8, and should not have an encoding declaration.

- All identifiers in the Python standard library MUST use ASCII-only identifiers, and SHOULD use English words wherever feasible (Open source projects with a global audience are encouraged to adopt a similar policy.)

## Imports

- **Imports should usually be on separate lines**

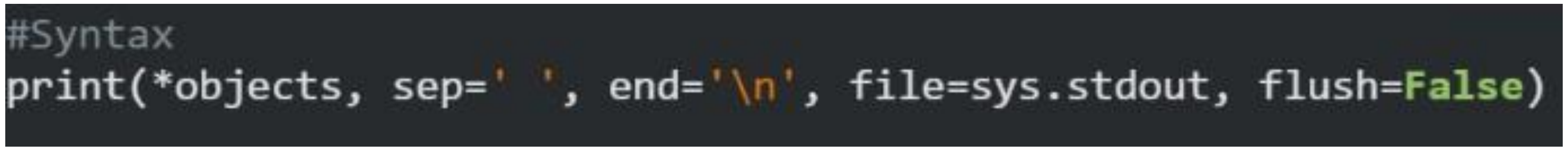
- **Imports are always put at the top of the file, just after any module comments and docstrings, and before module globals and constants.**

```
# Correct:
import os
import sys
```

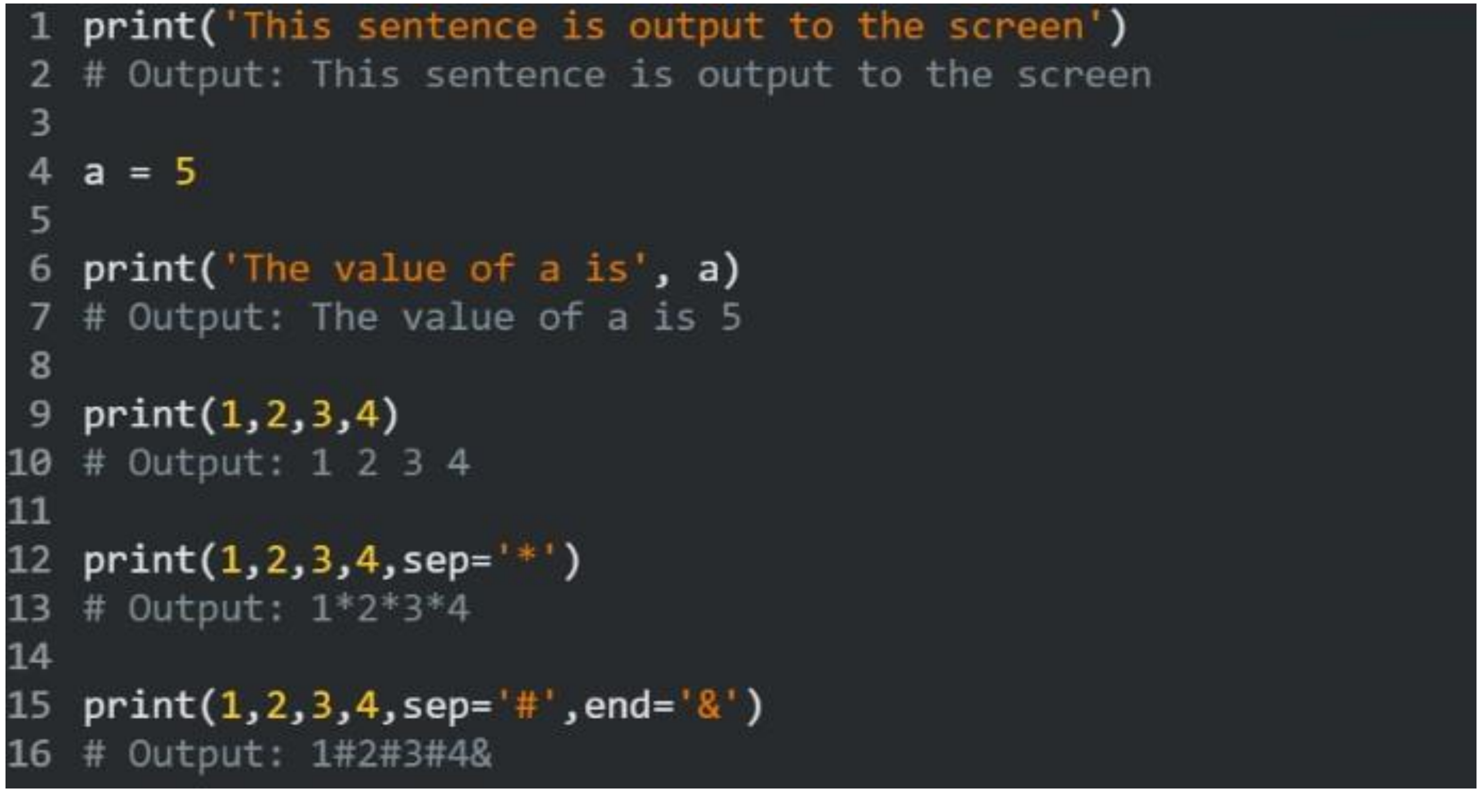
```
# Correct:
from subprocess import Popen, PIPE
```

# Python Input, Output

Python has the **print()** function to output data to the standard output device (screen):



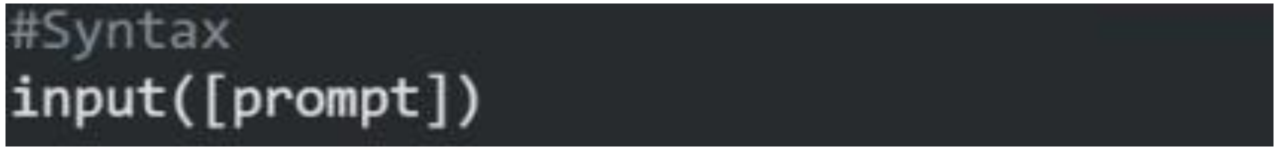
```
#Syntax
print(*objects, sep=' ', end='\n', file=sys.stdout, flush=False)
```



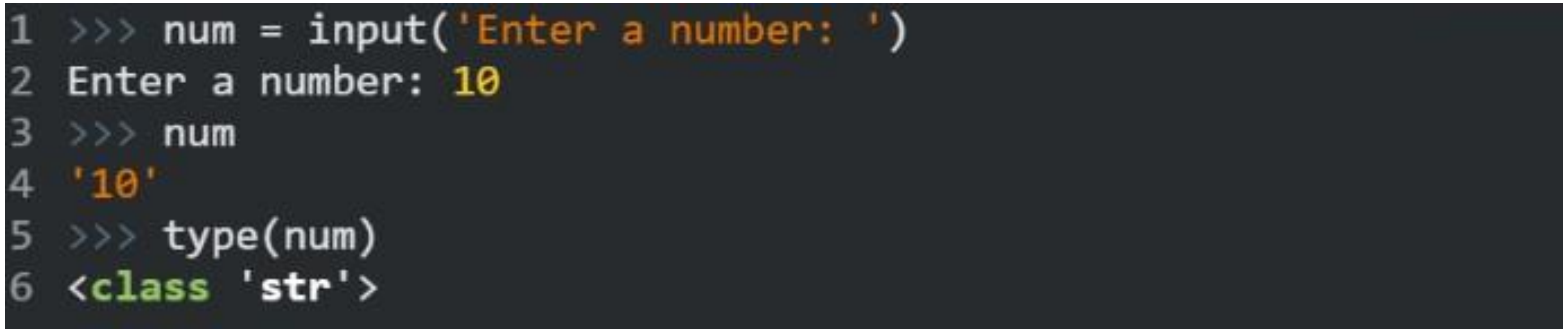
```
print('This sentence is output to the screen')
# Output: This sentence is output to the screen

a = 5

print('The value of a is', a)
# Output: The value of a is 5

print(1,2,3,4)
# Output: 1 2 3 4

print(1,2,3,4,sep='*')
# Output: 1*2*3*4

print(1,2,3,4,sep='#',end='&')
# Output: 1#2#3#4&
```

## Python Input, Output

In Python, we have the **input()** function to take the input from the user:



```
#Syntax
input([prompt])
```



```
>>> num = input('Enter a number: ')
Enter a number: 10
>>> num
'10'
>>> type(num)
<class 'str'>
```

**Style Guide for Python Code**

https://www.python.org/dev/peps/pep-0008/

**Writing user-friendly code with PEP-8 naming conventions**

https://www.youtube.com/watch?v=Sm0wwmEwqpI

**Register on the GitHub**

**Install Git Client on your computer**

## More questions …

softserve

Thank you!

TO
THE
FUTURE
softserve