Lecture 5 – Introduction to Python

What is Python Programming - The Basics

- Python is a general-purpose language.

- The syntax of the language is clean, and length of the code is relatively short. **It's fun to work in Python because it allows you to think about the problem rather than focusing on the syntax.**

History of Python

- Python is a fairly an old language created by Guido Van Rossum. The design began in the late 1980s and was first released in February 1991.

Why the name Python?

- No. It wasn't named after a dangerous snake. Rossum was **fan of a comedy series** from late seventies. The name "Python" was adopted from the same comedy series **"Monty Python's Flying Circus".**

Features of Python Programming

A simple language which is easier to learn

- Python has a very **simple and elegant syntax**. It's much easier to read and write Python programs compared to other languages like: C++, Java, C#. Python makes programming fun and allows you to focus on the solution rather than syntax.

Free and open source

- You can **freely use** and distribute Python, even for commercial use. Not only can you use and distribute software written in it, but you can also even make changes to the Python's source code.

Portability

- You **can move Python programs from one platform to another** and run it without any changes.
- It runs seamlessly on almost all platforms including Windows, Mac OS X and Linux.

Extensible and Embeddable

- Suppose an application requires high performance. You can easily combine pieces of C/C++ or other languages with Python code.

- This will give your application high performance as well as scripting capabilities which other languages may not provide out of the box.

A high-level, interpreted language

- Unlike C/C++, **you don't have to worry** about daunting tasks like **memory management, garbage collection** and so on.

- Likewise, when you run Python code, it automatically converts your code to the language your computer understands. **You don't need to worry about any lower-level operations.**

Object-Oriented

- Everything in **Python is an object**. Object oriented programming (OOP) helps you solve a complex problem intuitively.

- With OOP, you can **divide these complex problems into smaller sets** by creating objects

Applications of Python

Web Applications

- You can create scalable **Web Apps** using frameworks and CMS (Content Management System) that are built on Python.

- **Sites like Mozilla**, Reddit, Instagram, and PBS are written in Python.

Scientific and Numeric Computing

- **There are numerous libraries available in Python for scientific and numeric computing**. There are libraries like: SciPy and NumPy that are used in general purpose computing.

- And there are **specific libraries like**: EarthPy for earth science, AstroPy for Astronomy and so on.

- Also, the language is **heavily used** in machine learning, data mining and deep learning.

Good Language to Teach Programming

- Python is used by many companies to teach programming.

- It is a good language with a lot of features and capabilities. Yet, it's one of the **easiest languages** to learn because of its simple easy-to-use syntax.

Reasons to Choose Python as First Language

Simple Elegant Syntax

- Programming in Python is fun. It's easier to understand and write Python code. **Why?** The syntax feels natural.

Not overly strict

- You **don't need to define the type of a variable** in Python. Also, it's **not necessary to add semicolon** at the end of the statement.

- Python enforces you to follow good practices (**like proper indentation)**. These small things can make learning much easier for beginners.

Expressiveness of the language

- Python allows you to write programs having greater functionality with fewer lines of code.

Great Community and Support

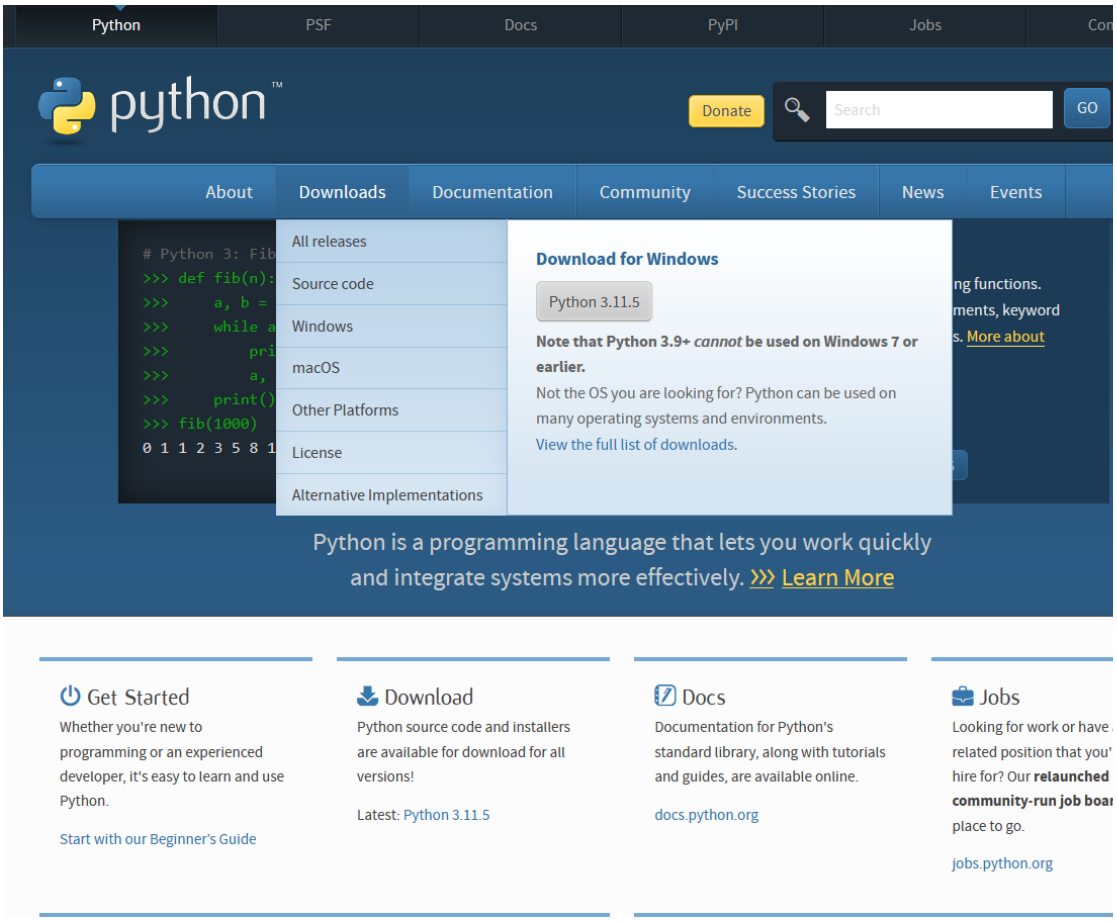
- Python has a large supporting community.

Python Installation

First software is : Python 3.11.5

To Download Python follow the following link :

https://www.python.org/



And the Second Software is :

JetBrains PyCharm

https://www.jetbrains.com/pycharm/download/#section=windows

Download and install.

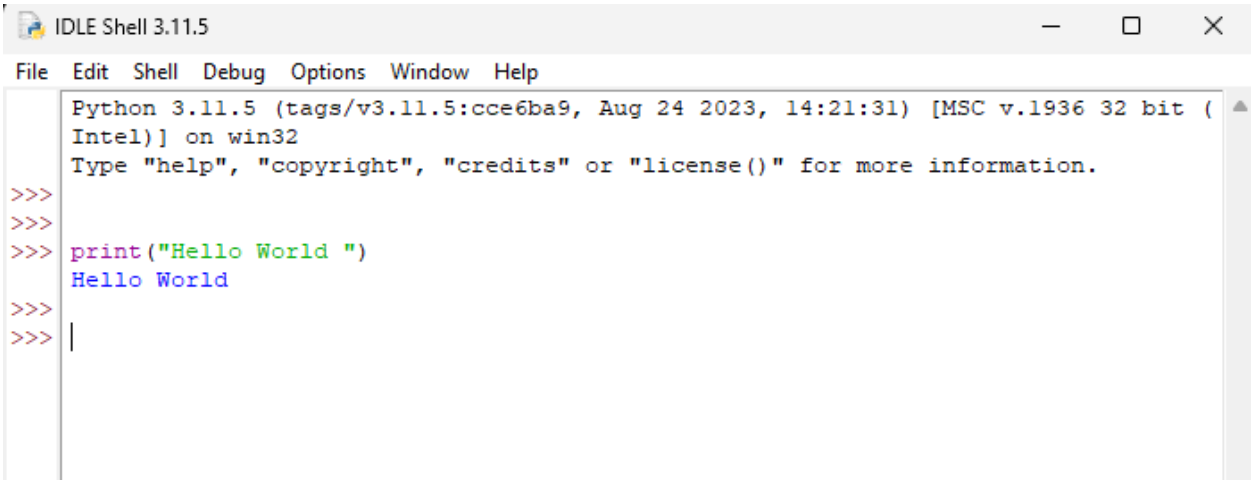
To write a program using Python 3.11.5 , follow the following steps

Start Python 3.11.5

Start write your code such as

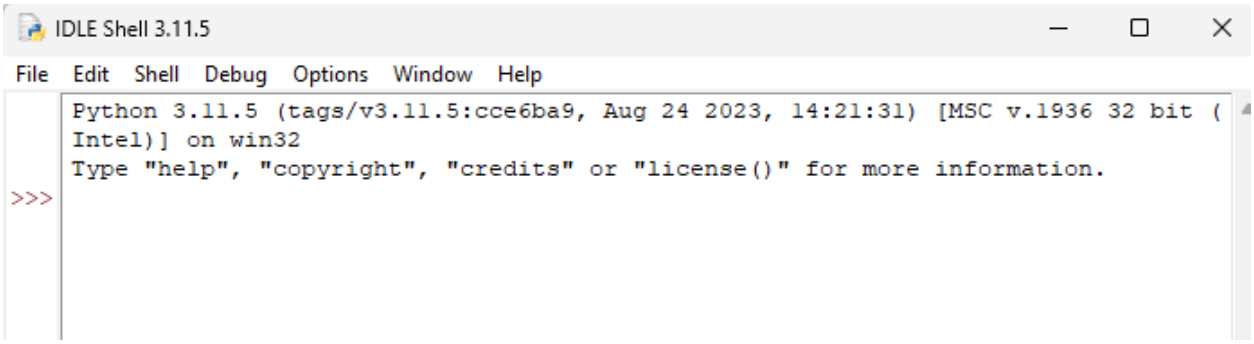
print("Hello World ")

and press enter



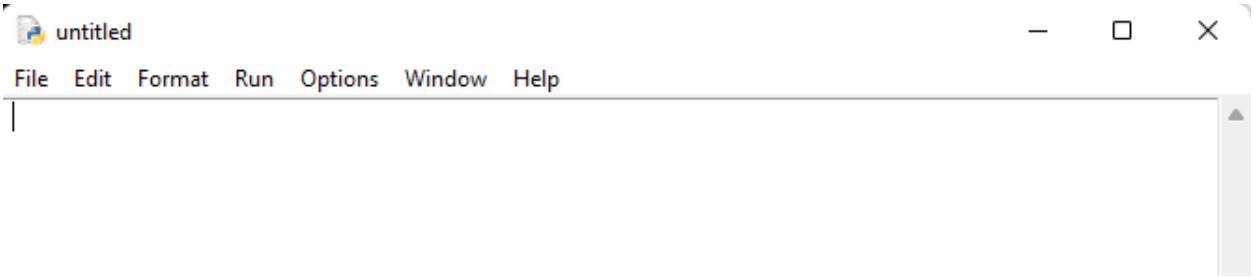
To create a program in a file using Python 3.11.5 , follow the following steps

Start Python 3.11.5

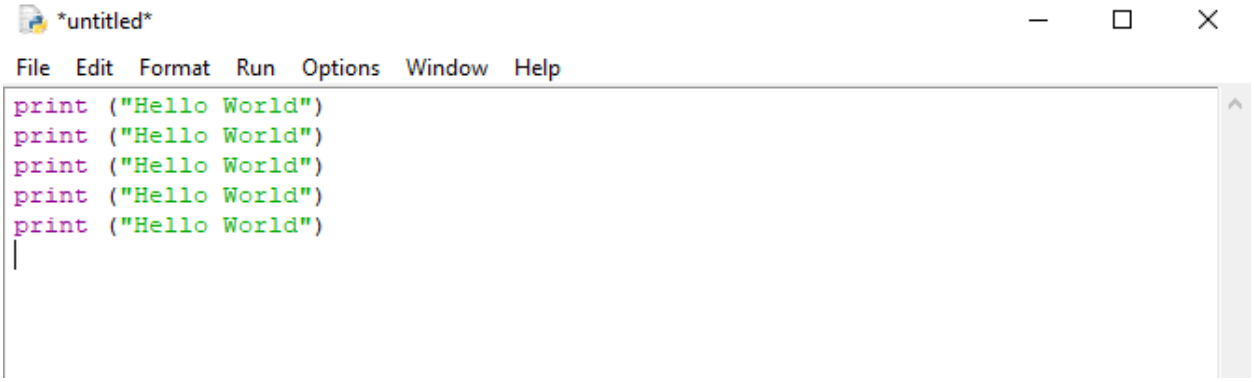


File → new file →

Your get a blank screen



Type your code



Then select file → save as → Select a Directory → write a name of the file.

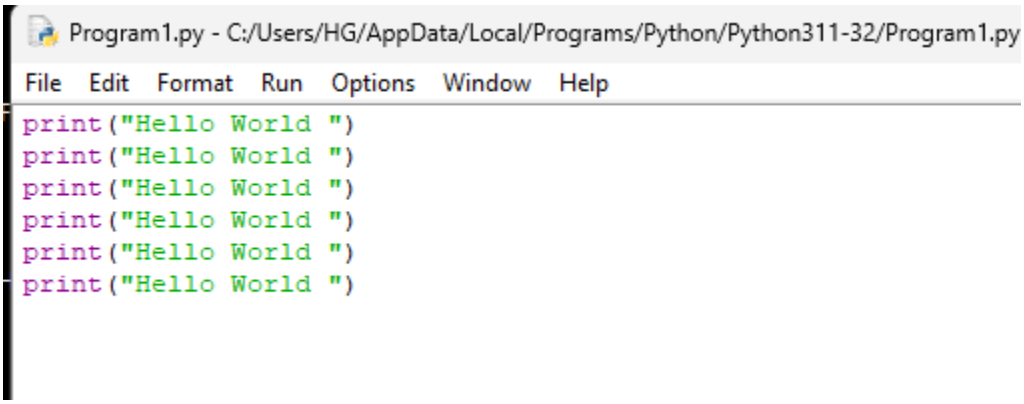
It must end with xxxxx.py

For example Program1.py

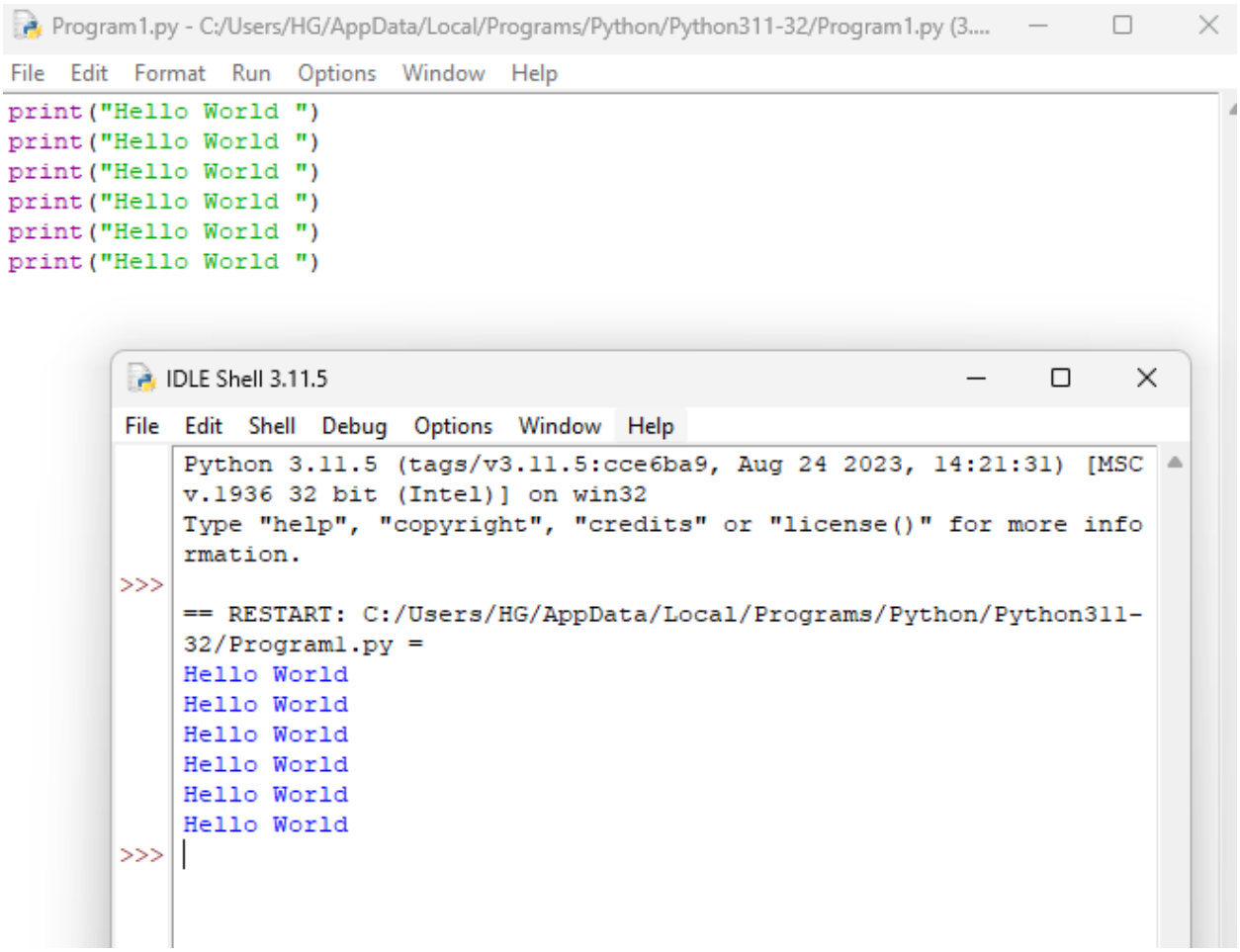
Press save.

To run the program

Then select form the menu Run → Run Module



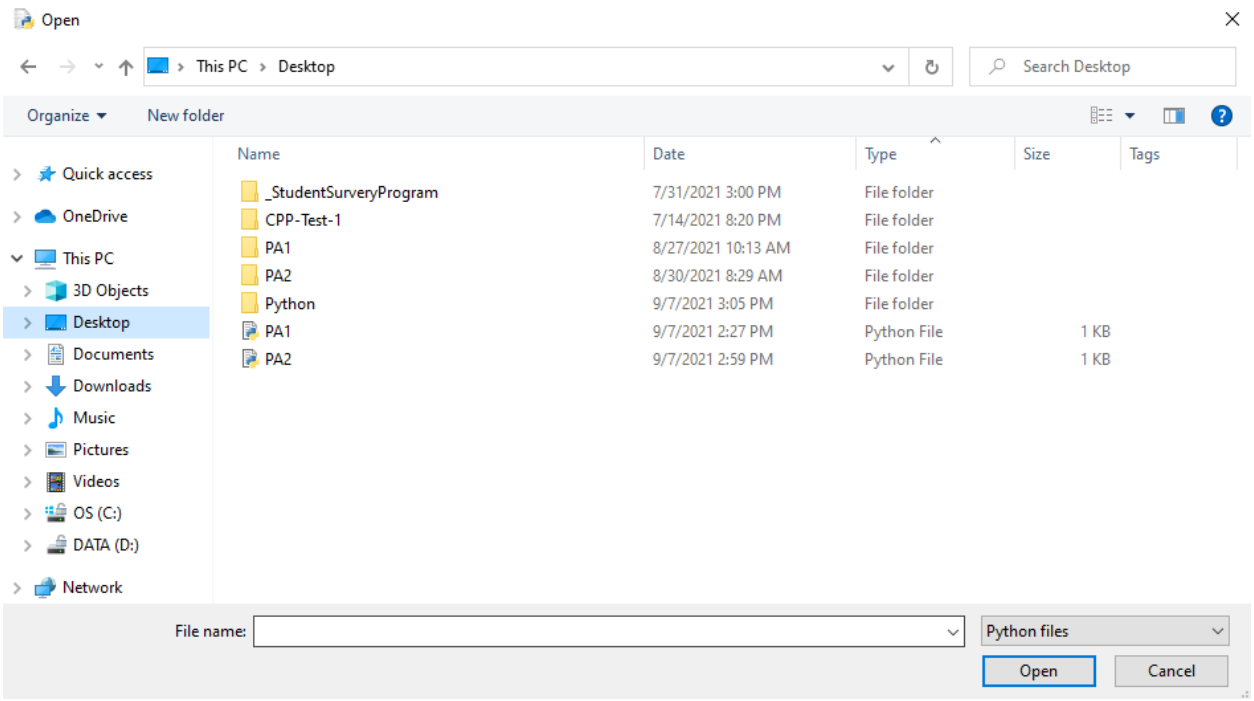
The output will be as follows



File exit will terminate the IDE

To open an existing file just select file → open

Then select the name of the from displayed directory and select the open button.



How To Write Your First Python 3 Program

The “Hello, World!” program is a classic and time-honored tradition in computer programming. Serving as a simple and complete first program for beginners, as well as a good program to test systems and programming environments, **“Hello, World!” illustrates the basic syntax of programming languages**.

To write the “Hello, World!” program, let’s open up a command-line text editor such as Python 3.11.1 or PyCharm and create a new file:

File → new file → and start typing your code

```
print("Hello World")
```

Then select save as and give it a name such as

HelloWorld.py

Finally, select file → Run Module

Hello World will be displayed

Let’s break down the different components of the code.

- print() is a function that tells the computer to perform an action. We know it is a function because it uses parentheses

- print() tells Python to display or output whatever we put in the parentheses. By default, this will output to the current terminal window.

Some functions, **like the print() function**, are **built-in** functions included in Python by default. These built-in functions are always available for us to use in programs that we create

We can also define our **own functions** that we construct ourselves through other elements

Inside the parentheses of the print() function is a **sequence of characters** — Hello, World! — that is enclosed in quotation marks.

Any characters that are inside of quotation marks are called a **string**.

In this example, the **string** Hello, World! is also **called an argument** since it is a value that is passed to a function.

The **quotes** that are on either side of Hello, World! **were not printed** to the screen because they are used to tell Python that they contain a string.

The **quotation marks** delineate where the string begins and ends.

Escape Characters

An *escape character* lets you use characters that are otherwise impossible to put into a string. An escape character consists of a backslash **(\)** followed by the character you want to add to the string.

For example, the escape character for a **single quote is** \'. You can use this inside a string that begins and ends with single quotes. To see how escape characters work, enter the following into the interactive shell:

Example

print("Let**\'**s print out this string.")

Let's print out this string.

>>>

The following are lists of the escape characters you can use

\' Single quote

\" Double quote

\n New Line

\t Tab

**** Backslash

Example

```
print('Hello there!')
print('How are you?')
print('I\'m \tdoing \tfine.')
```

Output

```
Hello there!
How are you?
I'm     doing      fine.
>>>
```

OR

```
print("Hello there!\nHow are you?\nI\'m \tdoing \tfine.")
```

Output

```
Hello there!
How are you?
I'm    doing    fine.
>>>
```

How To Write Comments

Comments are lines that exist in computer programs that are **ignored** by compilers and interpreters.

Including comments in programs **makes code more readable** for humans as it provides some information or explanation about what each part of a program is doing.

Comment Syntax

Comments in Python begin with a hash mark **(#)** and whitespace character and continue to the end of the line. Generally, comments will look something like this:

 # This is a comment

Example

```
# Print “Hello, World!” to console
print("Hello, World!")
```

Block Comments

Block comments can be used to explain more complicated code or code that you don't expect the reader to be familiar with.

In block comments, each line begins with the hash mark and a single space.

If you need to use more than one paragraph, they should be separated by a line **that contains a single hash mark**.

Example

```
# The main function will parse arguments via the parser variable.  These
# arguments will be defined by the user on the console.  This will pass
# the word argument the user wants to parse along with the filename the
# user wants to use, and also provide help text if the user does not
# correctly pass the arguments.
```

Inline Comments

Inline comments occur on the same line of a statement, following the code itself. Like other comments, they begin with a hash mark and a single whitespace character.

```
print("Hello World")    # Inline comment about the code
```

Multiple Lines

To create strings that span multiple lines, **triple single quotes** ''' or **triple double quotes** """ are used to enclose the string.

```
'''
This string is on
multiple lines
within three single
quotes on either side.
'''

"""
This string is on
multiple lines
within three double
quotes on either side.
"""
```

Keywords in Python

Keywords are the **reserved** words in Python.

We cannot use a keyword as variable name, function name or any other identifier.

Here's a list of all keywords in Python Programming

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

The above keywords may get altered in different versions of Python. Some extra might get added or some might be removed.

You can always get the list of keywords in your current version by typing the following in the prompt.

```
>>> import keyword

>>> print(keyword.kwlist)


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