

# Python Programming



1

# Introduction

- ▮ Introduction
- ▮ Why Programming?
- ▮ Programming for everybody.
- ▮ Any Prerequisites.
- ▮ Why do we have so many languages?
- ▮ Introduction to **Scratch**.

# Why to Learn Python?

- ▮ Python is **Open Source** which means its available free of cost.
- ▮ Python is **simple** and so **easy to learn**
- ▮ Python is **versatile** and can be **used to create many different things.**
- ▮ Python has **powerful development libraries** include **AI, ML** etc.
- ▮ Python is **much in demand** and **ensures high salary.**

# Getting Started with PYTHON

## ❖ What is PYTHON:

- ❖ Invented by **Guido van Rossum** in the **1990s**.
- ❖ Named after the comedy group Monty Python (popular BBC comedy show of that time, "**Monty Python's Flying Circus**".)
- ❖ **Open source** from the beginning
- ❖ Managed by **Python software foundation (PSF)**
- ❖ Aim - intelligent and could be taught quickly and effectively.



# Python Features:

## Easy to Learn

Few Keywords, simple structure  
and  
clear defined syntax

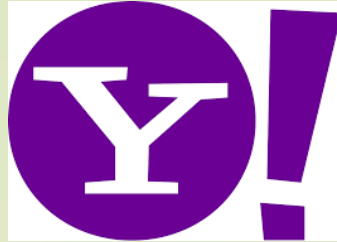
## Rapid Development

Let you get more done with less code and  
less time

## Powerful Standard library

Strength of Python is the bulk of the library is portable  
and cross platform compactable

# Who use Python today ?



# Python vs other popular language

Python	C/C++	JAVA	PHP
Faster to learn	Faster	Faster	
Portable code			
Much smaller and compact code			
More powerful standard library			
More versatile			Huge installed base

# Python vs other popular languages

C Program	Java Program	Python Program
<pre>main() {     printf("hello, world\n"); }</pre>	<pre>class myfirstjavaprogram {     public static void main(String args[])     {         System.out.println( "Hello World!");     } }</pre>	<pre>print ("Hello World!!")</pre>



# Python vs other popular language

- Python is a **cross-platform programming language**, which means that it can run on multiple platforms like Windows, macOS, Linux.
- It is **free and open-source**.
- Even though most of today's Linux and Mac have Python pre-installed in it, **the version might be out-of-date**. So, it is always a good idea to install the most current version.

# Python vs other popular language

- Python was designed for readability, and has some similarities to the English language with influence from mathematics.
- Python **uses new lines to complete a command**, as opposed to **other programming languages** which often use **semicolons or parentheses**.
- Python **relies on indentation, using whitespace, to define scope**; such as the scope of loops, functions and classes. Other programming languages often use **curly-brackets** for this purpose.

# The Easiest Way to Run Python:

- ▮ The easiest way to run Python is by using **Thonny IDE**.
- ▮ The Thonny IDE comes with the latest version of Python bundled in it. So we don't have to install Python separately.
- ▮ Steps to run Python on your computer.
  - ▮ Download Thonny IDE.
  - ▮ Run the installer to install Thonny on your computer.
  - ▮ Go to: File > New. Then save the file with .py extension. For example, hello.py, example.py, etc.
  - ▮ You can give any name to the file. However, the file name should end with .py
  - ▮ Write Python code in the file and save it.

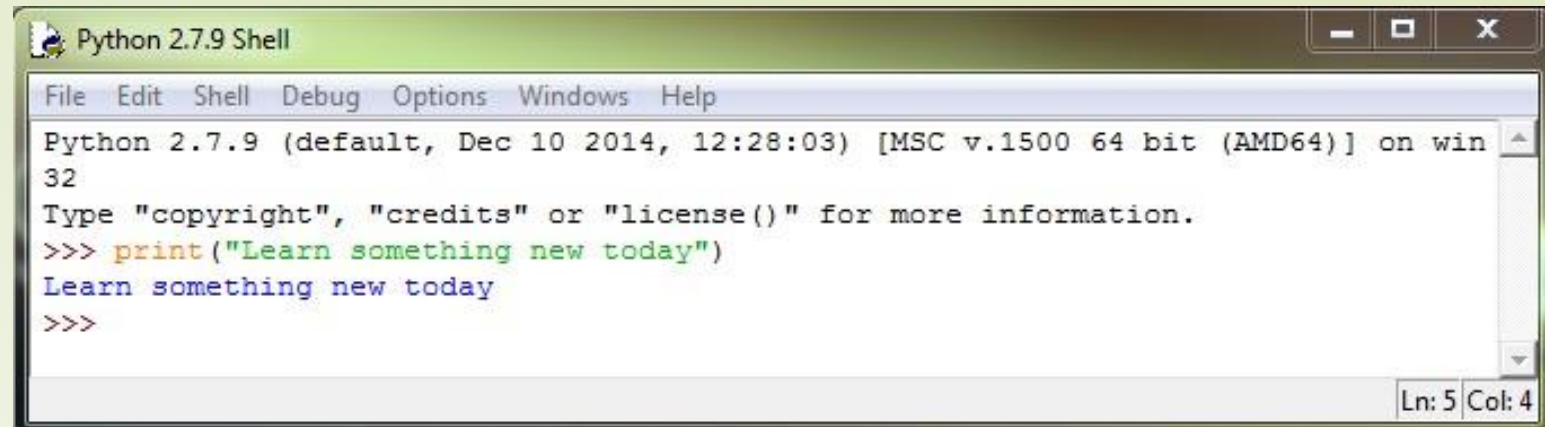
## Install Python Separately:

1. Download the [latest version of Python](#).
2. Run the installer file and follow the steps to install Python. During the install process, check **Add Python to environment variables**. This will add Python to environment variables, and you can run Python from any part of the computer. Also, you can choose the path where Python is installed.
3. Once you finish the installation process, you can run Python.

Once Python is installed, typing python in the command line will invoke the interpreter in immediate mode. We can directly type in Python code, and press Enter to get the output.

# Python CLE and IDLE Editor

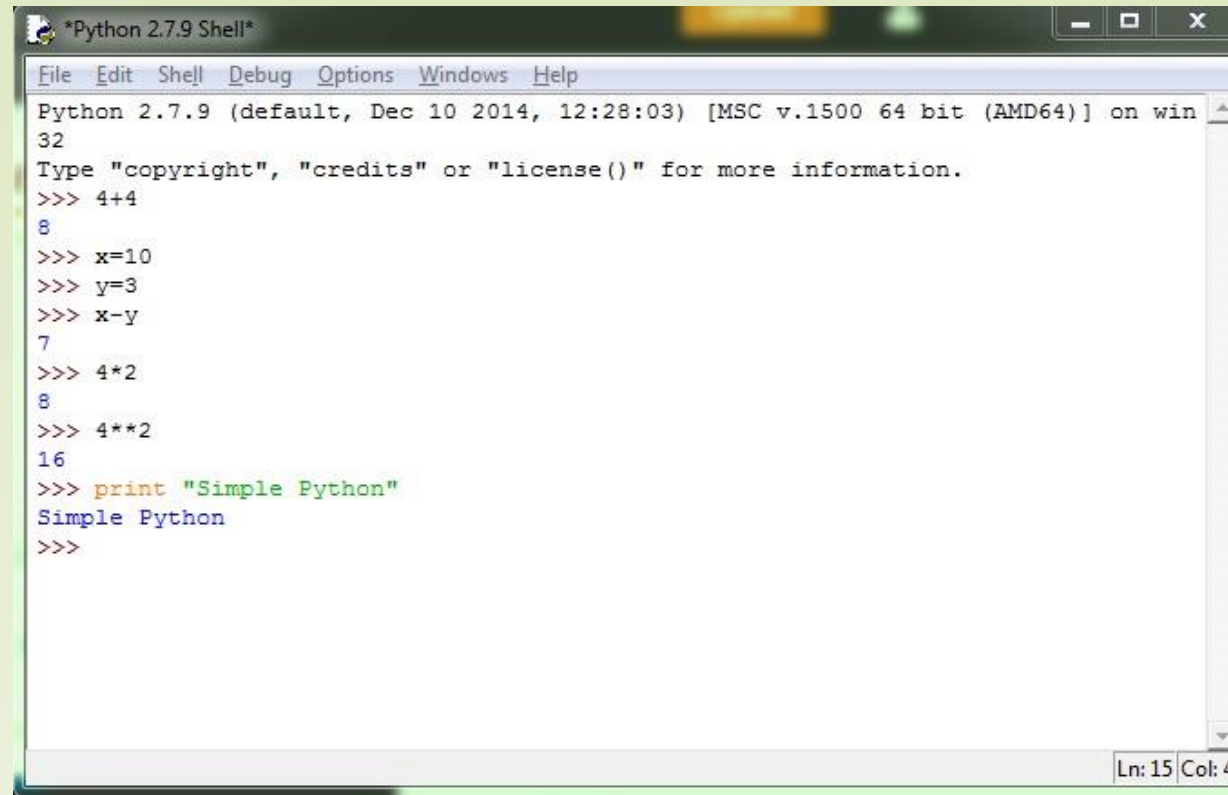
- ▮ **IDLE** (**I**ntegrated **D**evelopment **L**anguage **E**nvironment)
- ▮ Completely written in Python and the [Tkinter](#) GUI toolkit



```
Python 2.7.9 Shell
File Edit Shell Debug Options Windows Help
Python 2.7.9 (default, Dec 10 2014, 12:28:03) [MSC v.1500 64 bit (AMD64)] on win
32
Type "copyright", "credits" or "license()" for more information.
>>> print("Learn something new today")
Learn something new today
>>>
```

Ln: 5 Col: 4

# Python Interpreter



```
*Python 2.7.9 Shell*
File Edit Shell Debug Options Windows Help
Python 2.7.9 (default, Dec 10 2014, 12:28:03) [MSC v.1500 64 bit (AMD64)] on win
32
Type "copyright", "credits" or "license()" for more information.
>>> 4+4
8
>>> x=10
>>> y=3
>>> x-y
7
>>> 4*2
8
>>> 4**2
16
>>> print "Simple Python"
Simple Python
>>>
```

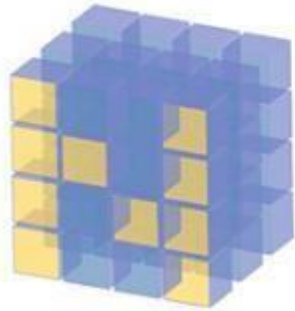
The screenshot shows a Windows-style application window titled "\*Python 2.7.9 Shell\*". The window has a menu bar with "File", "Edit", "Shell", "Debug", "Options", "Windows", and "Help". The main text area displays the Python 2.7.9 shell prompt and several lines of code and output. The code includes arithmetic operations (4+4, 4\*2, 4\*\*2, x-y) and a print statement. The output shows the results of these operations and the printed string "Simple Python". The status bar at the bottom right indicates "Ln: 15 Col: 4".

# How to Install Anaconda on Windows?

- ▮ **Anaconda** is an **open-source software** that contains **Jupyter, spyder, etc** that are used for large data processing, data analytics, heavy scientific computing.
- ▮ Anaconda **works for R and python** programming language. **Spyder(sub-application of Anaconda) is used for python.**
- ▮ Package versions are managed by the package management system called conda.
- ▮ For more details refer:  
▮ <https://www.geeksforgeeks.org/how-to-install-anaconda-on-windows/?ref=lbp>

# Awesome Standard Library

matplotlib



NumPy

Pandas





# Introduction

- Python is an interpreted programming language. Python source code are compiled to byte code as **.pyc** file, and this byte code can be interpreted.
- There are two modes for using the Python interpreter:
  - Interactive Mode**
  - Script Mode**

# Interactive Mode

- Without passing python script file to interpreter, directly execute code to Python prompt.
- Example:
  - `>>>2+6`
- Output:
  - `8`
- The chevron in the beginning of the 1st line, i.e. the symbol `>>>` is a prompt the python interpreter uses to indicate that it is ready. If the programmer types `2+6`, the interpreter replies `8`.

# Script Mode

Alternatively, programmers can store Python script source code in a file with **.py** extension, and use the interpreter to execute the contents of the file. To execute the script by the interpreter, you have to tell the interpreter the name of the file. For example, if you have a script name **MyFile.py** and you're working on Unix, to run the script you have to type:

```
python MyFile.py
```

# Modes Usage

- Working with interactive mode is better when Python programmers deal with small pieces of code as you can type and execute them immediately, but when the code is more than 2-4 lines, using the script for coding can help to modify and use the code in future.

# Python Keywords and Identifiers:

- ❑ The Python Keywords must be in your information because you can not use them as variable name or any other identifier name.
- ❑ Keywords are **predefined, reserved words** used in Python programming that have special meanings to the compiler.
- ❑ We **cannot use a keyword as a variable name, function name, or any other identifier**. They are used to define the syntax and structure of the Python language.
- ❑ All the keywords except **True, False and None** are in **lowercase** and they must be written as they are. The list of all the keywords is given below.

# Python Keywords and Identifiers:

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

# Python Identifiers:

- Identifiers are the name given to variables, classes, methods, etc. For example,

```
language = 'Python'
```

```
continue = 'Python'
```

- The above code is wrong because we have used `continue` as a variable name.

## Rules for Naming an Identifier:

- ▮ Identifiers cannot be a keyword.
- ▮ Identifiers are case-sensitive.
- ▮ It can have a sequence of letters and digits. However, it must begin with a letter or `_`. The first letter of an identifier cannot be a digit.
- ▮ It's a convention to start an identifier with a letter rather `_`.
- ▮ Whitespaces are not allowed.
- ▮ We cannot use special symbols like `!`, `@`, `#`, `$`, and so on.



# Rules for Naming an Identifier:

Valid Identifiers	Invalid Identifiers
score	@core
return_value	return
highest_score	highest score
name1	lname
convert_to_string	convert to_string

## Rules for Naming an Identifier:

- ❑ **Python is a case-sensitive language.** This means, **Variable** and **variable** are not the same.
- ❑ **Always give the identifiers a name that makes sense.** While `c = 10` is a valid name, writing `count = 10` would make more sense, and it would be easier to figure out what it represents when you look at your code after a long gap.
- ❑ **Multiple words can be separated using an underscore,** like `this_is_a_long_variable`.

# Python Comments:

- ❑ In computer programming, **comments are hints** that we use to make our code more understandable.
- ❑ Comments are completely ignored by the interpreter.
- ❑ Make Code Easier to Understand.
- ❑ Using Comments for Debugging.

```
# declare and initialize two variables
num1 = 6
num2 = 9

# print the output
print('This is output')
```

```
name = 'Eric Cartman' # name is a string
```

```
print('Python')

# print('Error Line ')

print('Django')
```

## Multi-line Comment in Python:

- Python doesn't offer a separate way to write multiline comments. However, there are other ways to get around this issue.

```
# This is a long comment  
# and it extends  
# to multiple lines
```

- Another way of doing this is to use triple quotes, either `'''` or `"""`.

```
''' This is also a  
perfect example of  
multi-line comments '''
```

# Python Comments:

- ▮ A comment is used to enter freehand text, usually for documentary purposes
- ▮ Use a single hashmark to comment #
- ▮ Comment ends with new line
  - ▮ # this is a comment
  - ▮ def sum(): # this is the sum function

Python Comments

```
1 price = 10.2
2 # increase price to 5%
3 price = price * 1.05
4
5 salary = 5000
6 salary = salary * 1.02 # increase salary 2% for the employee
7
8
9 def increase_salary(sal, rating, percentage):
10     """ increase salary base on rating and percentage
11     rating 1 - 2 no increase
12     rating 3 - 4 increase 5%
13     rating 4 - 6 increase 10%
14     """
15
```

# Python Variables:

- ▮ In programming, a variable is a container (storage area) to hold data.

```
number = 10
```

- ▮ We use the assignment operator = to assign a value to a variable.
- ▮ **Note: Python is a type-inferred language, so you don't have to explicitly define the variable type.**
- ▮ Changing the Value of a Variable in Python.
- ▮ Assigning multiple values to multiple variables

```
a, b, c = 5, 3.2, 'Hello'  
  
print(a) # prints 5  
print(b) # prints 3.2  
print(c) # prints Hello
```

```
a=1  
print(a)  
a=5  
print(a)
```

# Python Variables:

- ▮ A variable is created the moment you first assign a value to it.
- ▮ Variables **do not need to be declared with any particular type**, and can even change type after they have been set.
- ▮ If you want **to specify the data type** of a variable, this can be done with **casting**.
- ▮ Python allows you to **assign values to multiple variables** in one line.
- ▮ **(Note: Make sure the number of variables matches the number of values, or else you will get an error.)**

```
str1 = "SVNIT Surat"  
str1 = 10  
print(str1)
```

```
x = str(3)  
y = int(3)  
z = float(3)  
print(x)  
print(y)  
print(z)
```

```
x, y, z = "Orange", "Banana", "Cherry"  
print(x)  
print(y)  
print(z)
```

# Python Variables:

- ❑ We can assign the same value to multiple variables in one line.
- ❑ If you have a collection of values in a list, tuple etc. **Python allows you to extract the values into variables.** This is called unpacking.
- ❑ Variables that are created outside of a function are known as **global variables**.
- ❑ Global variables can be used by everyone, both inside of functions and outside.

```
str1 = str2 = "SVNIT Surat"  
print(str1)  
print(str2)
```

```
fruits = ["apple", "banana", "cherry"]  
x, y, z = fruits  
print(x)  
print(y)  
print(z)
```

```
x = "awesome"  
  
def myfunc():  
    x = "fantastic"  
    print("Python is " + x)  
  
myfunc()  
  
print("Python is " + x)
```



## Python Output Variables:

- ❑ The Python **print()** function is often used to output variables.
- ❑ In the `print()` function, output multiple variables, separated by a comma.
- ❑ You can also use the `+` operator to output multiple variables.
- ❑ In the `print()` function, when you try to combine a string and a number with the `+` operator, Python will give you an error

```
x = "Python is awesome"  
print(x)
```

```
x = "Python"  
y = "is"  
z = "awesome"  
print(x, y, z)
```

```
x = "Python "  
y = "is "  
z = "awesome"  
print(x + y + z)
```

```
x = 5  
y = "John"  
print(x + y)
```

## Python Variables:

- ❑ If you create a variable with the same name inside a function, this variable will be **local**, and **can only be used inside the function**. The global variable with the same name will remain as it was, global and with the original value.
- ❑ Normally, when you create a variable inside a function, that variable is local, and can only be used inside that function.
- ❑ **To create a global variable inside a function, you can use the global keyword.**
- ❑ To change the value of a global variable inside a function, refer to the variable by using the global keyword.

```
def myfunc():  
    global x  
    x = "fantastic"  
  
myfunc()  
  
print("Python is " + x)
```

```
x = "awesome"  
  
def myfunc():  
    global x  
    x = "fantastic"  
  
myfunc()  
  
print("Python is " + x)
```

# Rules for Naming Python Variables:

- Constant and variable names should have a combination of letters in lowercase (a to z) or uppercase (A to Z) or digits (0 to 9) or an underscore (\_). For example:

```
snake_case  
MACRO_CASE  
camelCase  
CapWords
```

```
MyVariableName = "John"
```

```
myVariableName = "John"
```

- Create a name that makes sense. For example, vowel makes more sense than v.
- If you want to create a variable name having two words, use underscore to separate them. For example:

```
my_name  
current_salary
```

- Python is case-sensitive. So num and Num are different variables.
- Avoid using keywords like if, True, class, etc. as variable names.

# Rules for Naming Python Variables:

- ▮ A variable can have a short name (like `x` and `y`) or a more descriptive name (`age`, `carname`, `total_volume`). Rules for Python variables:
  - ▮ A variable name must start with a letter or the underscore character
  - ▮ A variable name cannot start with a number
  - ▮ A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and `_`)
  - ▮ Variable names are case-sensitive (`age`, `Age` and `AGE` are three different variables)
  - ▮ A variable name cannot be any of the Python keywords.

# Quick Review:

- ❑ Create a variable named carname and assign the value Ferrari to it.
- ❑ Create a variable named x and assign the value 50 to it.
- ❑ Display the sum of 5 + 10, using two variables: x and y.
- ❑ Create a variable called z, assign x + y to it, and display the result.
- ❑ Remove the illegal characters in the variable name: 2my-first\_name = "SVNIT"
- ❑ Give correct syntax to assign the same value to all three variables in one code line.
- ❑ Insert the correct keyword to make the variable x belong to the global scope.

```
def myfunc():  
     x  
    x = "fantastic"
```

# Python Constants:

- ▮ A constant is a special type of variable whose value cannot be changed.
- ▮ In Python, constants are usually declared and assigned in a module (a new file containing variables, functions, etc. which is imported to the main file).

```
# declare constants
PI = 3.14
GRAVITY = 9.8
```

```
# import constant file we created above
import constant

print(constant.PI) # prints 3.14
print(constant.GRAVITY) # prints 9.8
```

# Python Data Types:

- ▮ Data types specify the type of data that can be stored inside a variable.
- ▮ You can get the data type of any object by using the `type()` function.

Data Types	Classes	Description
Numeric	int, float, complex	holds numeric values
String	str	holds sequence of characters
Sequence	list, tuple, range	holds collection of items
Mapping	dict	holds data in key-value pair form
Boolean	bool	holds either <code>True</code> or <code>False</code>
Set	set, frozenset	hold collection of unique items

# Python Data Types:

- Since everything is an object in Python programming, data types are actually classes and variables are instances(object) of these classes.

Example	Data Type
<code>x = "Hello World"</code>	str
<code>x = 20</code>	int
<code>x = 20.5</code>	float
<code>x = 1j</code>	complex
<code>x = ["apple", "banana", "cherry"]</code>	list
<code>x = ("apple", "banana", "cherry")</code>	tuple
<code>x = range(6)</code>	range
<code>x = {"name" : "John", "age" : 36}</code>	dict
<code>x = {"apple", "banana", "cherry"}</code>	set
<code>x = frozenset({"apple", "banana", "cherry"})</code>	frozenset



# Python Numeric Data Type:

- ▮ **Integers, floating-point numbers and complex numbers** fall under Python numbers category. They are defined as **int**, **float** and **complex** classes in Python.
  - ▮ **int** - holds signed integers of non-limited length.
  - ▮ **float** - holds floating decimal points and it's accurate up to 15 decimal places.
  - ▮ **complex** - holds complex numbers.
- ▮ We can use the **type() function** to know which class a variable or a value belongs to.

## Python Numeric Data Type:

```
@author: Balu Laxman
"""
num1 = 5
print(num1, 'is of type', type(num1))

num2 = 2.0
print(num2, 'is of type', type(num2))

num3 = 1+2j
print(num3, 'is of type', type(num3))
```

```
5 is of type <class 'int'>
2.0 is of type <class 'float'>
(1+2j) is of type <class 'complex'>
```

# Python Numbers, Type Conversion and Mathematics:

- ▮ The **number data types** are used **to store the numeric values**.
- ▮ Python supports integers, floating-point numbers and complex numbers. They are defined as `int`, `float`, and `complex` classes in Python.
  - ▮ **int** - holds signed integers of non-limited length.
  - ▮ **float** - holds floating decimal points and it's accurate up to 15 decimal places.
  - ▮ **complex** - holds complex number

# Python Numeric Data Type:

- Integers and floating points are separated by the presence or absence of a decimal point. For instance,
  - 5 is an integer
  - 5.42 is a floating-point number.
- Complex numbers are written in the form,  $x+yj$ , where  $x$  is the real part and  $y$  is the imaginary part.
- We can use the `type()` function to know which class a variable or a value belongs to.

```
num1 = 5
print(num1, 'is of type', type(num1))

num2 = 5.42
print(num2, 'is of type', type(num2))

num3 = 8+2j
print(num3, 'is of type', type(num3))
```

# Number Systems:

- ▮ The numbers we deal with every day are of the decimal (base 10) number system.
- ▮ But computer programmers need to work with binary (base 2), hexadecimal (base 16) and octal (base 8) number systems.
- ▮ In Python, we can represent these numbers by appropriately placing a prefix before that number. The following table lists these prefixes.

Number System	Prefix
Binary	0b or 0B
Octal	0o or 0O
Hexadecimal	0x or 0X

```
print(0b1101011) # prints 107  
print(0xFB + 0b10) # prints 253  
print(0o15) # prints 13
```

# Type Conversion in Python:

- ▮ In programming, type conversion is the process of converting one type of number into another.
- ▮ Operations like addition, subtraction convert integers to float implicitly (automatically), if one of the operands is float. For example,
- ▮ We can also use built-in functions like `int()`, `float()` and `complex()` to convert between types explicitly.

```
print(1 + 2.0) # prints 3.0
```

```
num1 = int(2.3)
print(num1)

num2 = int(-2.8)
print(num2)

num3 = float(5)
print(num3)

num4 = complex('3+5j')
print(num4)
```

# Python Random Module:

- Python offers the random module to generate random numbers or to pick a random item from an iterator.
- First we need to import the random module.

```
import random

print(random.randrange(10, 20))

list1 = ['a', 'b', 'c', 'd', 'e']

# get random item from list1
print(random.choice(list1))

# Shuffle list1
random.shuffle(list1)

# Print the shuffled list1
print(list1)

# Print random element
print(random.random())
```

```
12
c
['b', 'a', 'e', 'c', 'd']
0.8915375820057005
```

# Thank you