

# INTRODUCTION TO PYTHON

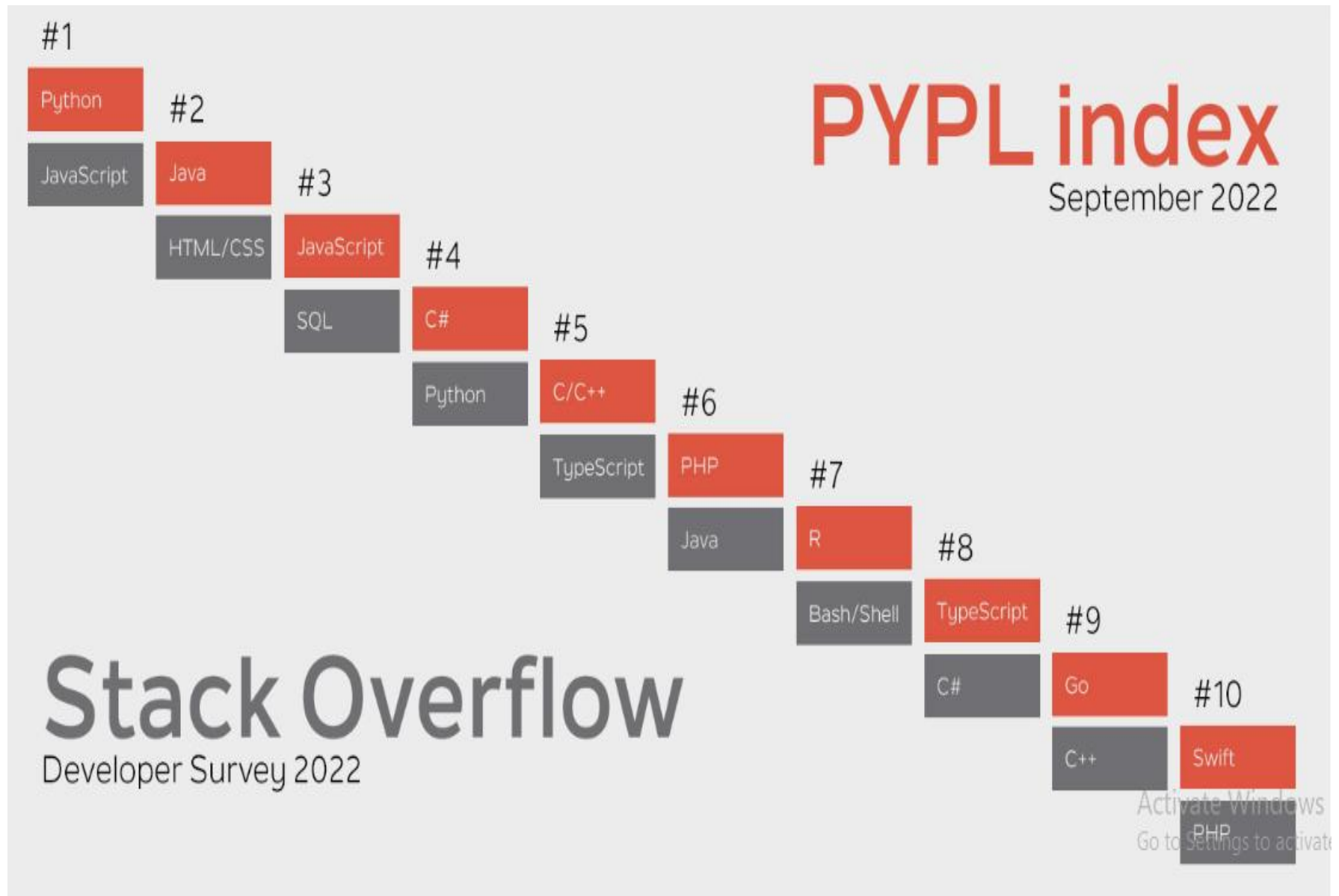
Python is an interpreted, object-oriented, high-level programming language with dynamic semantics developed by **Guido van Rossum**. It was originally released in **1991**.

## Python Use Cases

- Creating web applications on a server
- Building workflows that can be used in conjunction with software
- Connecting to database systems
- Reading and modifying files
- Performing complex mathematics
- Processing big data
- Developing production-ready software

# INTRODUCTION TO PYTHON

## Popularity of Programming Language Index:



# INTRODUCTION TO PYTHON

## Difference between Compiler and Interpreter

COMPILER	INTERPRETER
A compiler takes the entire program in one go.	An interpreter takes the single line of code at a time.
A compiler generates an intermediate machine code.	An interpreter never produces any intermediate code.
Error are displayed after entire program is checked. (If any)	Errors are displayed for every instruction is interpreted. (If any)
Memory requirements is more.	Memory requirement is less.
Examples: C, C++,C#, Java etc.	Examples: BASIC, Python, Perl, Ruby etc.

# INTRODUCTION TO PYTHON

## Some important reasons for popularity:

- **Easy to read and Code:** Python is a very high-level programming language, yet it is effortless to learn and learning the basic Python syntax is very easy, as compared to other popular languages like C, C++, and Java.
- **Free and Open source:** Python is developed under an OSI-approved open source license. Hence, it is completely free to use, even for commercial purposes.
- **Robust standard Library:** This means that programmers don't have to write their code for every single thing unlike other programming languages.
- **Interpreted language:** When a programming language is interpreted, it means that the source code is executed line by line, and not all at once. There is no need to compile Python because it is processed at runtime by the interpreter.
- **Portable:** Python is portable in the sense that the same code can be used on different machines.
- **Object-Oriented and Procedure-Oriented:** One of the critical Python features is that it supports both object-oriented and procedure-oriented programming.

# INTRODUCTION TO PYTHON

## Python History and Versions

- Python laid its foundation in the late 1980s.
- The implementation of Python was started in December 1989 by **Guido Van Rossum** at CWI in Netherland.
- In February 1991, **Guido Van Rossum** published the code (labeled version 0.9.0) to alt.sources.
- In January 1994, Python 1.0 was released with new features like lambda, map, filter, and reduce.
- In October 2000, Python 2.0 added new features such as list comprehensions, garbage collection systems.
- On December 2008, Python 3.0 (also called "Py3K") was released. It was designed to rectify the fundamental flaw of the language.
- *ABC programming language* is said to be the predecessor of Python language, which was capable of Exception Handling and interfacing with the Amoeba Operating System.
- **Latest version of Python is 3.11.3**

# INTRODUCTION TO PYTHON

## IDE (Integrated Development Environment Software)

An integrated development environment (IDE) is a **software application that helps programmers develop software code efficiently**. It increases developer productivity by combining capabilities such as software editing, building, testing, and packaging in an easy-to-use application.



IDLE

Python Software Foundation...



PyCharm

Apache License



Spyder

MIT License



Thonny

MIT License



Visual Studio

Proprietary software



Atom

MIT License



PyDev

Eclipse Public License



Wing IDE

Proprietary software



eric

GNU General Public License



Python Tools for Visual St...

Apache License



PyScripter

MIT License



Eclipse

Common Public License



# INTRODUCTION TO PYTHON

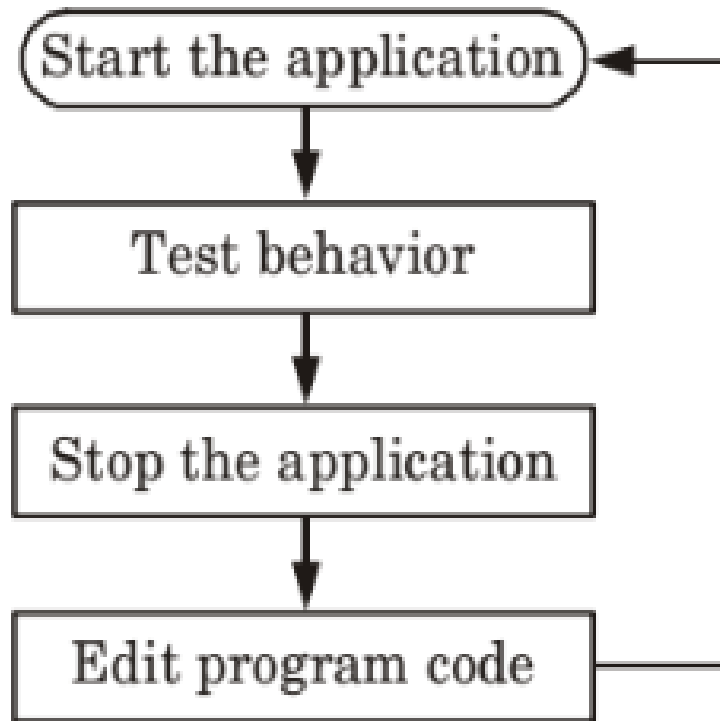
## Programming Cycle:

1. Python's programming cycle is dramatically shorter than that of traditional programming cycle.
2. In Python, there are no compile or link steps.
3. Python programs simply import modules at runtime and use the objects they contain. Because of this, Python programs run immediately after changes are made.
4. In cases where dynamic module reloading can be used, it is even possible to change and reload parts of a running program without stopping it at all.

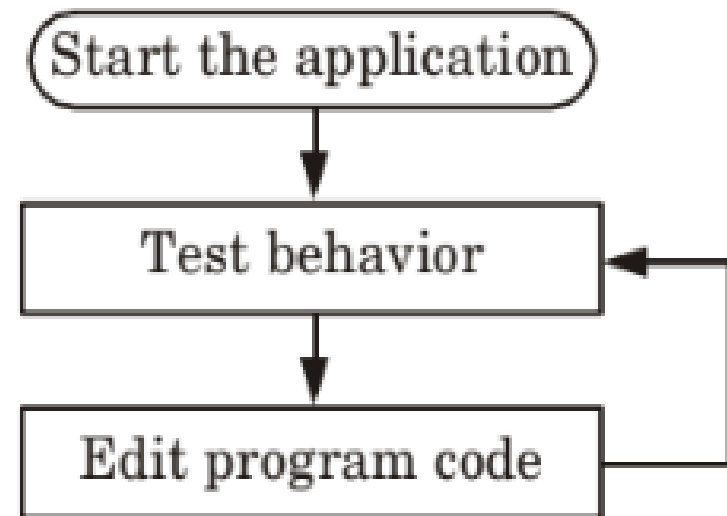
# INTRODUCTION TO PYTHON

## Programming Cycle (Cont...):

5. Python's impact on the programming cycle is as follows:



(a) Python's programming cycle



(b) Python's programming cycle with module reloading



# INTRODUCTION TO PYTHON

## Traditional Programming Cycle

The programmer writes the code, compiles it, and executes it.

The code is compiled into an executable file.

The programmer must declare variables before using them.

The programmer needs to specify the data type of the variables.

The programmer needs to free the memory allocated for the variables.

The programmer needs to handle memory management.

## Python Programming Cycle

The programmer writes the code and then runs it.

The code is directly interpreted by the Python interpreter.

The programmer does not need to declare variables before using them.

The programmer does not need to specify the data type of the variables.

The interpreter automatically releases the memory allocated for the variables.

The interpreter handles memory management.

# INTRODUCTION TO PYTHON

**Q: What makes python programming different from others?**

**Ans.** Python programming is different from other languages in a few key ways.

## **Interpreted language**

Python is an interpreted language, meaning that your code is not compiled into an executable program. Instead, the Python interpreter runs your code line by line, translating each line into machine code.

## **Dynamically type**

Python is a dynamically typed language, which means that you don't have to declare the types of your variables ahead of time.

## **High-level language**

Python is a very high-level language, which means that it abstracts away a lot of the low-level details that you would have to deal with in other languages.

# INTRODUCTION TO PYTHON

## **Python script:**

A Python script, the file containing the commands, is structured to be executed like a program. These files are designed to contain various functions and import various modules. Python interactive shell or the respective command line is used to execute the script, to perform a specific task.

## **Script:**

- It is a plain text file containing Python code.
- It has the extension **.py**
- It can be executed directly.
- Python interpreter is responsible for executing the Python scripts.

# INTRODUCTION TO PYTHON

## Different ways to run python script:

- Interactive mode
- Command Line
- Text Editor
- IDE

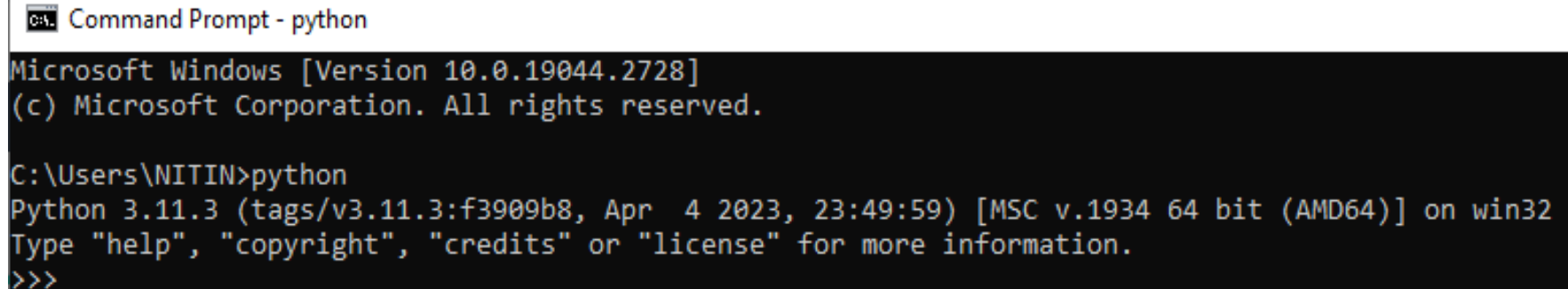
# INTRODUCTION TO PYTHON

## Different ways to run python script:

- Interactive mode

This is the most frequently used way to run a Python code. Here basically we do the following:

1. Open command prompt / terminal in your system
2. Enter the command - python or python3 (based on python version installed in your system)
3. If we see these : **>>>** then, we are into our python prompt. It will look something like this:



```
C:\> Command Prompt - python
Microsoft Windows [Version 10.0.19044.2728]
(c) Microsoft Corporation. All rights reserved.

C:\Users\NITIN>python
Python 3.11.3 (tags/v3.11.3:f3909b8, Apr  4 2023, 23:49:59) [MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

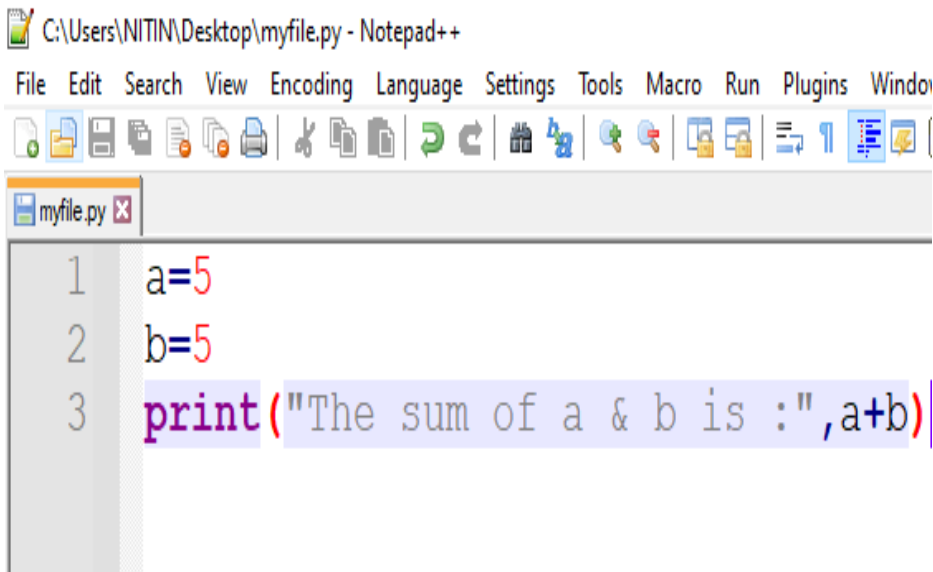
# INTRODUCTION TO PYTHON

## Different ways to run python script:

### •Command Line

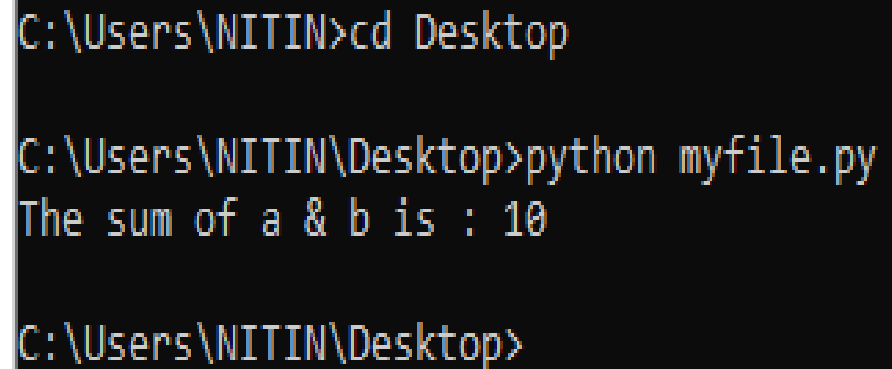
Follow the below steps to run a Python Code through command line -

1. Write your Python code into any text editor.
2. Save it with the extension .py into a suitable directory
3. Open the terminal or command prompt.
4. Type python/python3 (based on installation) followed by the file name.py –  
**python filename.py**



A screenshot of the Notepad++ text editor. The title bar shows the file path 'C:\Users\NITIN\Desktop\myfile.py' and the application name 'Notepad++'. The menu bar includes File, Edit, Search, View, Encoding, Language, Settings, Tools, Macro, Run, Plugins, and Window. The toolbar contains various icons for file operations and editing. The editor window shows a Python script with three lines: '1 a=5', '2 b=5', and '3 print("The sum of a & b is :",a+b)'. The third line is highlighted with a light blue background.

```
C:\Users\NITIN\Desktop\myfile.py - Notepad++  
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window  
1 a=5  
2 b=5  
3 print("The sum of a & b is :",a+b)
```



A screenshot of a Windows command prompt window with a black background and white text. It shows the directory being changed to the Desktop, then the execution of a Python script, and the resulting output.

```
C:\Users\NITIN>cd Desktop  
  
C:\Users\NITIN\Desktop>python myfile.py  
The sum of a & b is : 10  
  
C:\Users\NITIN\Desktop>
```

# INTRODUCTION TO PYTHON

## Different ways to run python script:

### •Run Python on Text Editor

If you want to run your Python code into a text editor -- suppose VS Code, then follow the below steps:

- 1.From extensions section, find and install 'Python' extension.
- 2.Restart your VS code to make sure the extension is installed properly.
- 3.Create a new file, type your Python code and save it.
- 4.Run it using the VS code's terminal by typing the command "py hello.py"
- 5.Or, Right Click -> Run Python file in terminal

# INTRODUCTION TO PYTHON

## Different ways to run python script:

### •Run Python on IDLE(IDE)

Python installation comes with an **Integrated Development and Learning Environment**, which is popularly known as IDLE.

### **Steps to run a python program on IDLE:**

- 1.Open the Python IDLE(navigate to the path where Python is installed, open IDLE(python version))
- 2.The shell is the default mode of operation for Python IDLE. So, the first thing you will see is the Python Shell –
- 3.You can write any code here and press enter to execute
- 4.To run a Python script you can go to **File -> New File**
- 5.Now, write your code and save the file –
- 6.Once saved, you can run your code from **Run -> Run Module** Or simply by **pressing F5** --



# Python Syntax

## Execute Python Syntax

As we learned in the previous page, Python syntax can be executed by writing directly in the Command Line:

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

Or by creating a python file on the server, using the .py file extension, and running it in the Command Line:

```
C:\Users\Your Name>python myfile.py
```

# Python Syntax

## Python Indentations

Where in other programming languages the indentation in code is for readability only, in Python the indentation is very important.

Python uses indentation to indicate a block of code.

Example:- `if 5 > 2:`

```
    print("Five is greater than two!")
```

Python will give you an error if you skip the indentation:

Example

```
if 5 > 2:
```

```
print("Five is greater than two!")
```



# Python Comment

- Comments can be used to explain Python code.
- Comments can be used to make the code more readable.
- Comments can be used to prevent execution when testing code.

## Creating a Comment

Comments starts with a #, and Python will ignore them:

Example

```
#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:

Example

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

# Python Comment

## 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:

Example

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

Or, not quite as intended, you can use a multiline string.

# Python Comment

## Multi Line Comments

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:

Example

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

As long as the string is not assigned to a variable, Python will read the code, but then ignore it, and you have made a multiline comment.

# Python USER INPUT

## User Input:

- Python allows for user input.
- That means we are able to ask the user for input.
- The method is a bit different in Python 3.X than Python 2.7.
- Python 3.6 uses the `input()` method.
- Python 2.7 uses the `raw_input()` method.

## Python 3.6

```
username = input("Enter username:")  
print("Username is: " + username)
```

## Python 2.7

```
username = raw_input("Enter username:")  
print("Username is: " + username)
```

# Python Variables

## Creating Variables

Variables are containers for storing data values.

Unlike other programming languages, Python has no command for declaring a variable.

A variable is created the moment you first assign a value to it.

## Example

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

# Python Variables

Variables do not need to be declared with any particular type and can even change type after they have been set.

## Example

```
x = 4 # x is of type int
x = "Sally" # x is now of type str
print(x)
```

String variables can be declared either by using single or double quotes:

## Example

```
x = "John"
# is the same as
x = 'John'
```



# Python Variables

## Variable Names

*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)
- \*Remember that variable names are case-sensitive**

# Python Variables

## Assign Value to Multiple Variables

Python allows you to assign values to multiple variables in one line:

Example

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

And you can assign the ***same*** value to multiple variables in one line:

Example

```
x = y = z = "Orange"  
print(x)  
print(y)  
print(z)
```

# Python Variables

## Output Variables

The Python *print* statement is often used to output variables. To combine both text and a variable, Python uses the **+** character:

### Example

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

You can also use the **+** character to add a variable to another variable:

### Example

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

# Python Variables

## Output Variables

For numbers, the **+** character works as a mathematical operator:

### Example

```
x = 5  
y = 10  
print(x + y)
```

**If you try to combine a string and a number, Python will give you an error:**

### Example

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

# Python Numbers

**There are three numeric types in Python:**

- int
- float
- complex

**Variables of numeric types are created when you assign a value to them:**

Example

```
x = 1    # int  
y = 2.8  # float  
z = 1+1j # complex
```

**To verify the type of any object in Python, use the type() function:**

Example

```
print(type(x))  
print(type(y))  
print(type(z))
```

# Python Numbers

## Int

Int, or integer, is a whole number, positive or negative, without decimals, of unlimited length.

Example

Integers:

```
x ,y, z = 1, 35656222554887711, 54646543
```

```
print(x)
```

```
print(y)
```

```
print(z)
```

# Python Numbers

## Float

Float, or "floating point number" is a number, positive or negative, containing one or more decimals.

Example

Floats:

```
x ,y, z = 1.0, 1.015, -0.25
```

```
print(x)  
print(y)  
print(z)
```

# Python Numbers

Float can also be scientific numbers with an "e" to indicate the power of 10.

Example

Floats:

```
x , y ,z= 35e3, 12E4, -87.7e100
```

```
print(x)
```

```
print(y)
```

```
print(z)
```



# Python Numbers

## Complex

Complex numbers are written with a "j" as the imaginary part:

Example

Complex:

```
x, y, z = 3+5j, 5j, -5j
```

```
print(x)
```

```
print(y)
```

```
print(z)
```

# Python Numbers

## Type Conversion

You can convert from one type to another with the **int()**, **float()**, and **complex()** methods:

### Example

Convert from one type to another:

```
x = 1 # int
```

```
y = 2.8 # float
```

```
z = 1j # complex
```

```
a = float(x) #convert from int to float
```

```
b = int(y) #convert from float to int
```

```
c = complex(x) #convert from int to complex
```

```
print(a)
```

```
print(b)
```

```
print(c)
```

```
print(type(a))
```

```
print(type(b))
```

```
print(type(c))
```

**Note: You cannot convert complex numbers into another number type.**

# Python Numbers

## Random Number

Python does not have a random() function to make a random number, but Python has a built-in module called random that can be used to make random numbers:

### Example

*Import the random module, and display a random number between 1 and 9:*

```
import random
```

```
print(random.randrange(1,10))
```

# Python Numbers

## Reserved Words (Keywords):

**Keywords in Python** are reserved words that can not be used as a variable name, function name, or any other identifier.

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

# Python Expression

A combination of operands and operators is called an **expression**. The expression in Python produces some value or result after being interpreted by the Python interpreter.

*Ex:*

```
x = 25      # a statement  
x = x + 10  # an expression  
print(x)
```

- An expression in Python can contain **identifiers**, **operators**, and **operands**.
- An **identifier** is a name that is used to define and identify a class, variable, or function in Python.
- An **operand** is an object that is operated on.
- An **operator** is a special symbol that performs the arithmetic or logical computations on the operands.

# Python Expression

## Types of Expression

There are various types of expression:

- **Constant Expressions:** A constant expression in Python that contains only constant values is known as a constant expression.

```
x = 10 + 15
```

```
print("The value of x is: ", x)
```

- **Arithmetic Expressions:** An arithmetic expression is a combination of numeric values, operators, and sometimes parenthesis. The result of this type of expression is also a numeric value.

```
x = 10 y = 5
```

```
addition = x + y
```

- **Integral Expressions:** These are the kind of expressions that produce only integer results after all computations and type conversions.

```
x = 10 y = 5.0
```

```
result = x + int(y)
```

# Python Expression

## Types of Expression (Cont...)

- Floating Expressions:** These are the kind of expressions which produce floating point numbers as result after all computations and type conversions.

```
x = 10 y = 5.0
```

```
result = float(x) + y
```

- Relational Expressions:** A relational expression in Python can be considered as a combination of two or more arithmetic expressions joined using relational operators. We have four types of relational operators in Python. (>, <, >=, <=).

```
x=10 y=5
```

```
result= x>y
```

- Logical Expressions:** As the name suggests, a logical expression performs the logical computation, and the overall expression results in either True or False (boolean result).

# Python Expression

## Types of Expression (Cont..)

- Bitwise Expressions:** The expression in which the operation or computation is performed at the bit level is known as a bitwise expression in Python.

```
x = 25
```

```
left_shift = x << 1
```

```
right_shift = x >> 1
```

- Combinational Expressions:** A combination expression can contain a single or multiple expressions which result in an integer or boolean value depending upon the expressions involved.

```
x = 25 y = 35
```

```
result = x + (y << 1)
```



# Python Assignment statements

We use Python assignment statements to assign objects to names. The target of an assignment statement is written on the left side of the equal sign (=), and the object on the right can be an arbitrary expression that computes an object.

**There are some important properties of assignment in Python :-**

- Assignment creates object references instead of copying the objects.
- Python creates a variable name the first time when they are assigned a value.
- Names must be assigned before being referenced.
- There are some operations that perform assignments implicitly.

# Python Assignment statements

## Different forms of assignment statements:

### Basic form:

```
st = 'abc'
```

```
print(st)
```

### Tuple assignment:

```
x, y = 50, 100
```

```
print('x = ', x)
```

```
print('y = ', y)
```

### List assignment:

```
[x, y] = [2, 4]
```

```
print('x = ', x)
```

```
print('y = ', y)
```

# Python Assignment statements

## Different forms of assignment statements:

### Sequence assignment:

```
a, b, c = 7, 5, 9
```

```
print('a = ', a)
```

```
print('b = ', b)
```

```
print('c = ', c)
```

### Multiple- target assignment:

```
x = y = 75
```

```
print(x, y)
```

### Augmented assignment :

```
x = 2
```

```
x += 1 # equivalent to: x = x + 1
```

```
print(x)
```

# Python Operators

**Python divides the operators in the following groups:**

- Arithmetic operators
- Assignment operators
- Comparison operators
- Logical operators
- Identity operators
- Membership operators
- Bitwise operators

# Python Operators (Arithmetic)

Operator	Name	Example
+	Addition	$x + y$
-	Subtraction	$x - y$
*	Multiplication	$x * y$
/	Division	$x / y$ (Result in float)
%	Modulus	$x \% y$
**	Exponentiation	$x ** y$
//	Floor division	$x // y$

# Python Operators (Assignment)

Operator	Example	Same As
=	x = 5	x = 5
+=	x += 3	x = x + 3
-=	x -= 3	x = x - 3
*=	x *= 3	x = x * 3
/=	x /= 3	x = x / 3
%=	x %= 3	x = x % 3
//=	x //= 3	x = x // 3
**=	x **= 3	x = x ** 3

# Python Operators (Comparison)

Operator	Name	Example
<code>==</code>	Equal	<code>x == y</code>
<code>!=</code>	Not equal	<code>x != y</code>
<code>&gt;</code>	Greater than	<code>x &gt; y</code>
<code>&lt;</code>	Less than	<code>x &lt; y</code>
<code>&gt;=</code>	Greater than or equal to	<code>x &gt;= y</code>
<code>&lt;=</code>	Less than or equal to	<code>x &lt;= y</code>

# Python Operators (Logical)

Operator	Description	Example
and	Returns True if both statements are true	<code>x &lt; 5 and x &lt; 10</code>
or	Returns True if one of the statements is true	<code>x &lt; 5 or x &lt; 4</code>
not	Reverse the result, returns False if the result is true	<code>not(x &lt; 5 and x &lt; 10)</code>



# Python Operators (Identity)

Identity operators are used to compare the objects, not if they are equal, but if they are actually the same object, with the same memory location:

Operator	Description	Example
is	Returns True if both variables are the same object	x is y
is not	Returns True if both variables are not the same object	x is not y

# PROGRAMS:

**PROGRAMS 1: WAP to find the type of variable.**

```
a=5
b="hello"
c=1.2
d=1+5j
print("The type of a is :",type(a))
print("The type of b is :",type(b))
print("The type of c is :",type(c))
print("The type of d is :",type(d))
```

## OUTPUT

```
The type of a is : <class 'int'>
The type of b is : <class 'str'>
The type of c is : <class 'float'>
The type of d is : <class 'complex'>
```

# PROGRAMS:

## PROGRAMS 2: WAP to find the square root of a number.

```
print("Enter the number for square root:: ")
num=float(input())
sqrt=num**0.5
print("The square root of the number is : ",sqrt)
```

## OUTPUT

```
Enter the number for square root::
16
The square root of the number is : 4.0
```