An Introduction to Python & Basic Python Syntax

By Aksadur Rahman

aksadur@yahoo.com

Agenda

A Brief History of Python Versions
Installing Python
Variables
Data Types
Dynamic Types
Python Reserved Words
Naming Conventions
Instruction/Statement
Basic Syntax Comments
Receiving Input
Type Conversion/Casting
Numeric Data Types
Boolean Data Types
Swapping
Strings

An Introduction to Python

An Introduction to Python

What is Python?

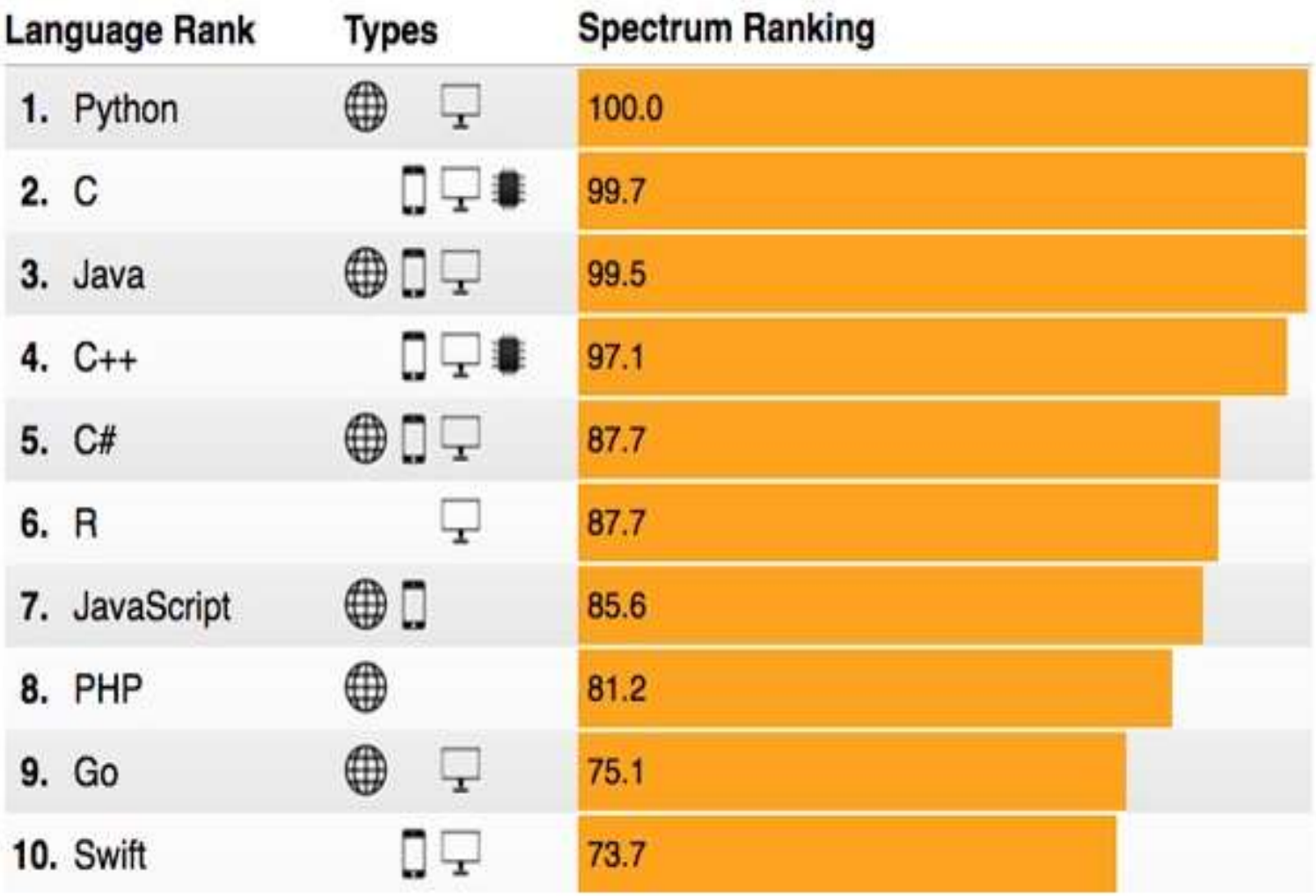
Python is a high-level object-oriented programming language that was created by Guido van Rossum. It is also called general-purpose programming language as it is used in almost every domain we can think of as mentioned below:

- ❑ Web Development
- ❑ Software Development
- ❑ Game Development
- ❑ AI & ML
- ❑ Data Analytics

An Introduction to Python

Why Python Programming?

IEEE spectrum list of top programming language 2021. The list of programming languages is based on popularity.



An Introduction to Python

Python is easy to understand

Java

```
class HelloWorld {
  static public void main( String args[] ) {
    System.out.println( "Hello World!" );
  }}
```

C++

```
#include <iostream.h>
main()
{
    cout << "Hello World!" << endl;
    return 0;
}
```

C#

```
class HelloWorld
{
    static void Main()
    {
        System.Console.WriteLine("Hello, World!");
    }
}
```

Python

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

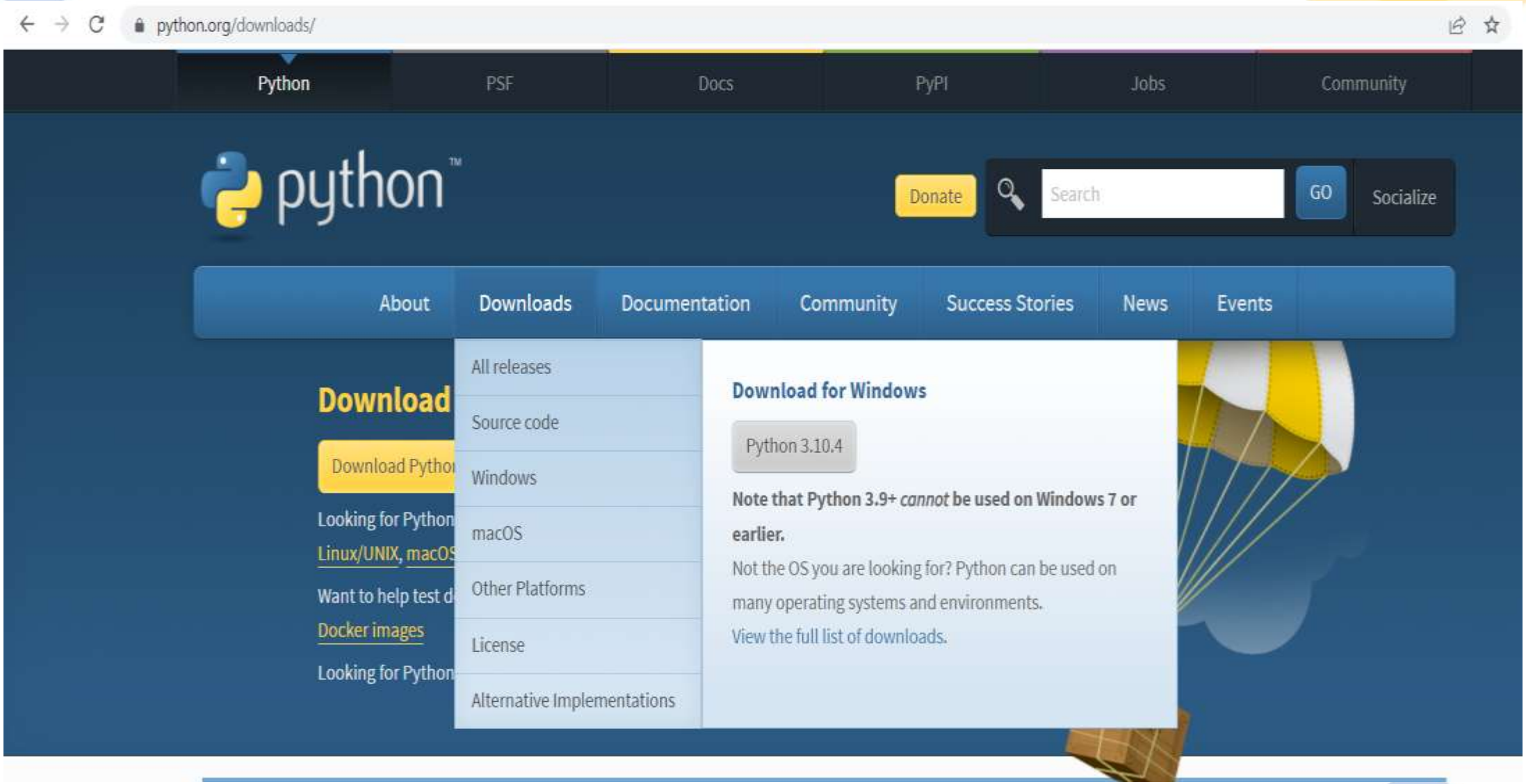
A Brief History of Python Versions

Python Version	Released Date
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

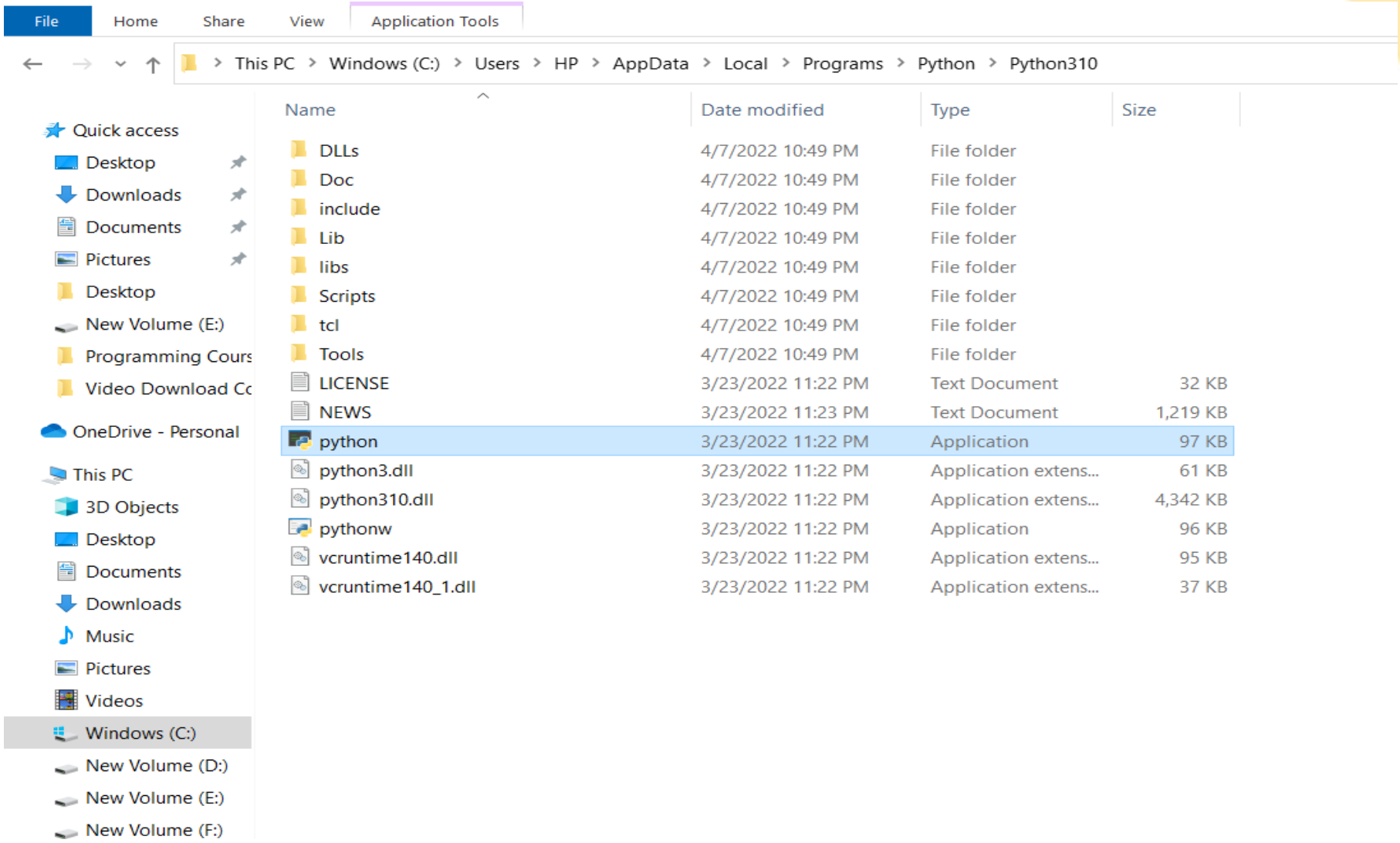
Python Software Foundation

First of all, there are the Pythons which are maintained by the people gathered around the PSF (Python Software Foundation), a community that aims to develop, improve, expand, and popularize Python and its environment. The PSF's president is Guido von Rossum himself

Installing Python



Installing Python



Installing Python

Installing PyCharm

jetbrains.com/pycharm/download/#section=windows

JET BRAINS

Developer Tools Team Tools Learning Tools Solutions Support Store

PyCharm

Coming in 2022.1 What's New Features Learn Pricing

Download PyCharm

Windows macOS Linux

Version: 2021.3.3
Build: 213.7172.26
17 March 2022

System requirements

Installation instructions

Other versions

Professional

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

Download

Free 30-day trial available

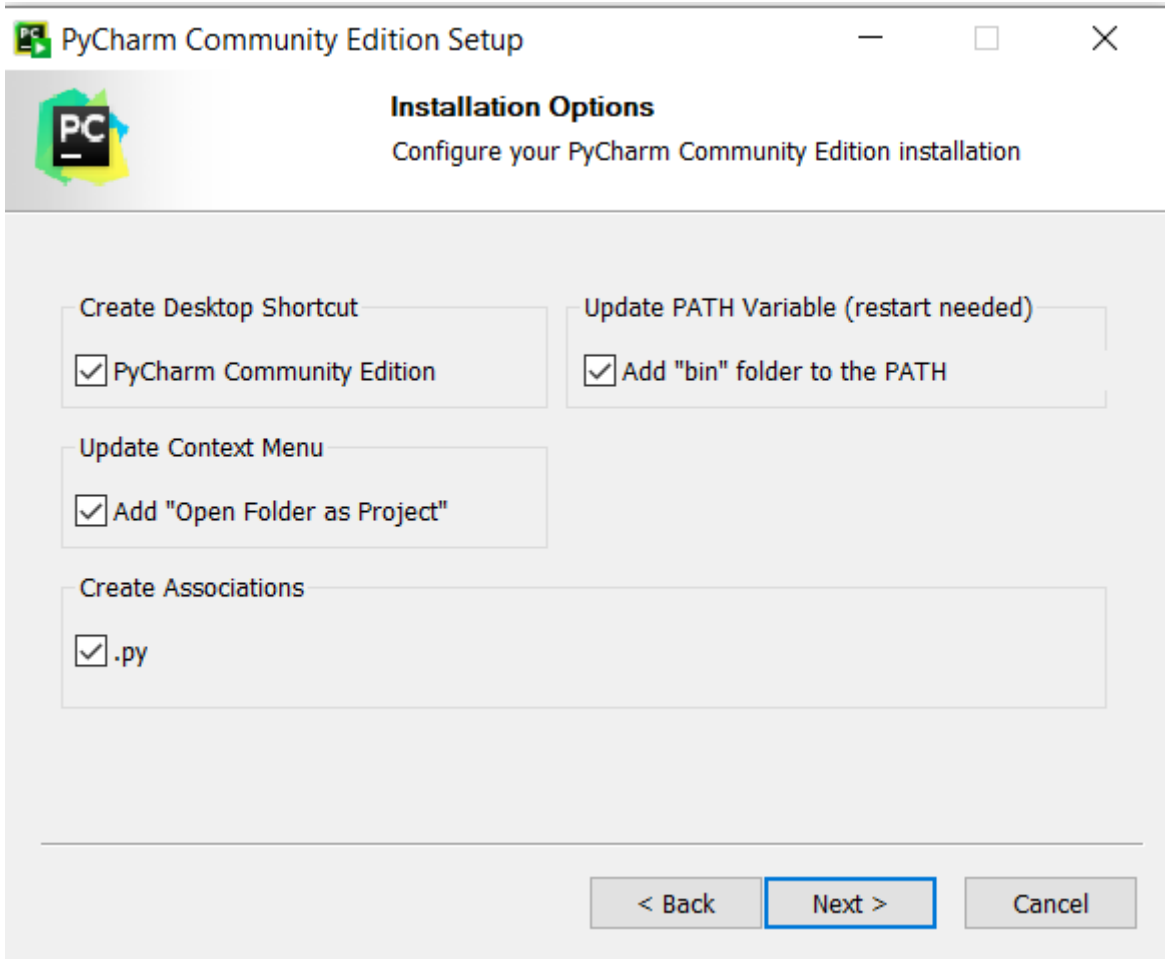
Community

For pure Python development

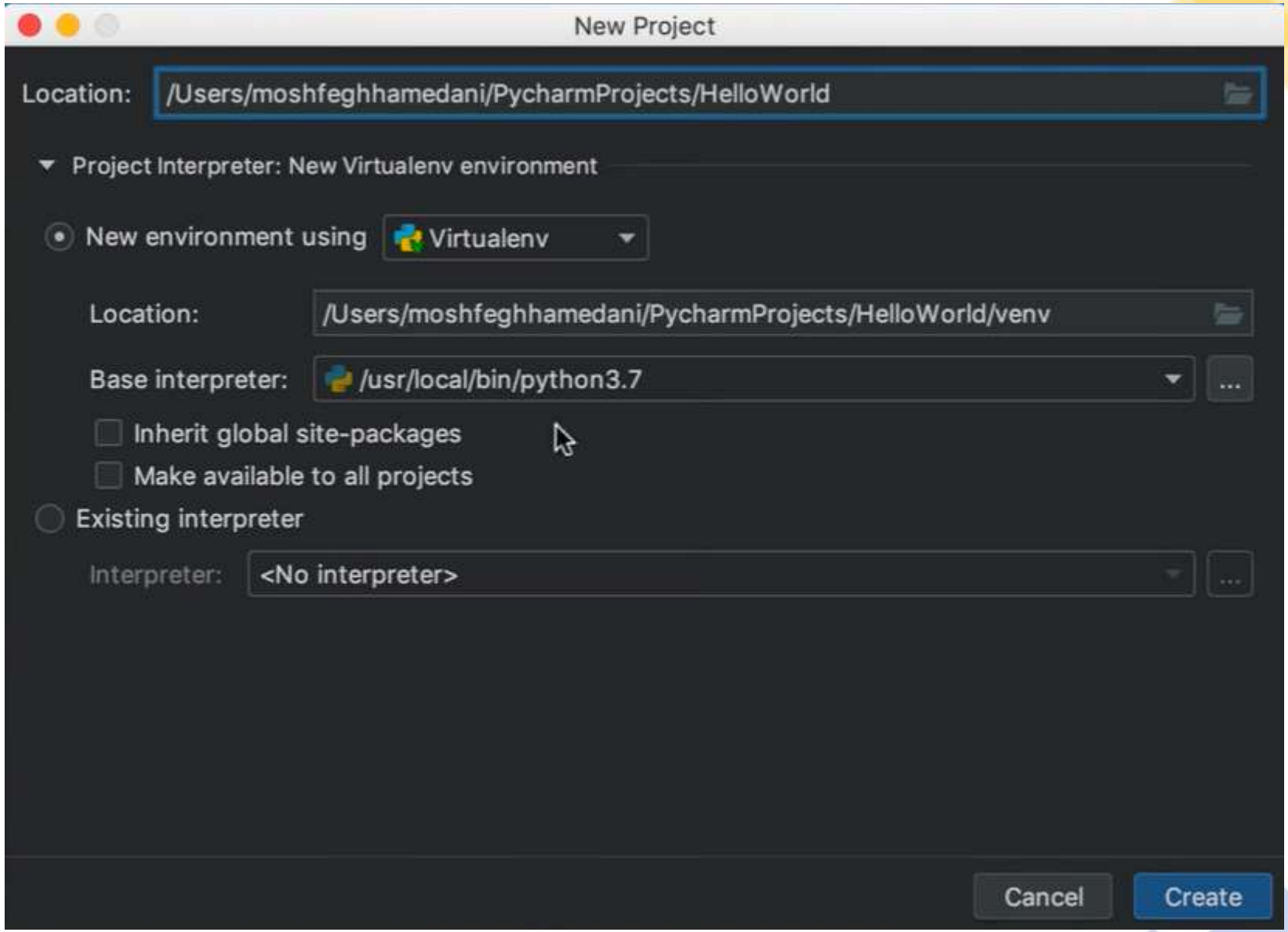
Download

Free, built on open-source

Installing PyCharm



Your First Python Program



Your First Python Program

File Edit View Navigate Code Refactor Run Tools VCS Window Help PythonCode - App.py

PythonCode App.py

App.py main.py

```
print("Bismillah")
```

```
D:\Python_Project\PythonCode\venv\Scripts\python.exe D:/Python_Project/PythonCode/App.py
Bismillah

Process finished with exit code 0
```

Download pre-built shared indexes: Reduce the indexing time and CPU load with pre-built Python packages shared indexes // Always download // Download once // Don't show again // Configure... (today 9:08 AM)

Variables

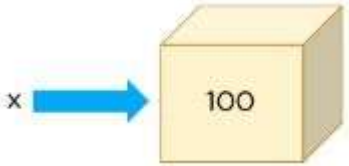
Variable is a name that is used to refer to memory location. It stores and manipulates data.

Variables are entities of a program that holds a value. Here is an example of a variable:

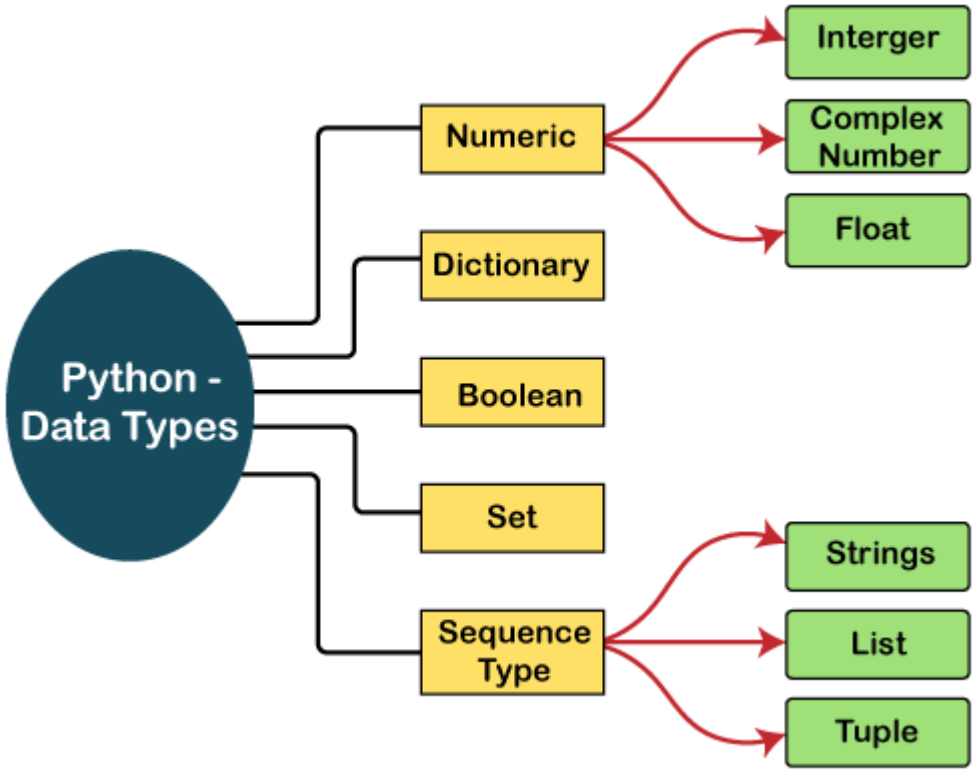
x=100

```
x=100

type(x)

int
```

In the below diagram, the box holds a value of 100 and is named as x. Therefore, the variable is x, and the data it holds is the value.



Data Types


Python - Data Types
Numeric
Dictionary
Boolean
Set
Sequence Type
Interger
Complex Number
Float
Strings
List
Tuple

Dynamic Types

Python is a **dynamically typed** language. It doesn't know about the type of the variable until the code is run.

```
x = 6

print(type(x))

x = 'hello'

print(type(x))
```

Naming Conventions

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:

- **Rule-1**: You should start variable name with an alphabet or underscore(_) character.
- **Rule-2**: A variable name can only contain **A-Z,a-z,0-9** and **underscore(_).**
- **Rule-3**: You cannot start the variable name with a number.
- **Rule-4**: You cannot use special characters with the variable name such as such as **$,%,#,&,@.-,^** etc.
- **Rule-5**: Variable names are case sensitive. For example **str and Str** are two different variables.
- **Rule-6**: Do not use reserve keyword as a variable name for example keywords like **class, for, def, del, is, else, try, from,** etc**.**

Python Reserved Keywords

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

Instruction/Statement

A **statement is an instruction that a Python interpreter can execute**. So, in simple words, we can say anything written in Python is a statement.

```
# Statement 1
print('Hello')

# Statement 2
x = 20

# Statement 3
print(x)
```

Basic Syntax Comments

```
print('Bismillah')

# print('Bismillah')

'''
print('Bismillah')
print('Bismillah')
'''


"""
print('Bismillah')
print('Bismillah')
"""
print('Alhamdulillah')
```

Receiving Input

Get user input with Python using the input() function. The user can enter keyboard input in the console

```
print("Enter your first name: ")

name = input()

print("Nice to meet you", name)
```

```
name = input("Enter your first name: ")

print("Nice to meet you", name)
```

Type Conversion/Casting

Type Conversion

The process of converting the value of one data type (integer, string, float, etc.) to another data type is called type conversion. Python has two types of type conversion.

- **Implicit Type Conversion**

  In Implicit type conversion, Python automatically converts one data type to another data type. This process doesn't need any user involvement.

```
num_int = 123
num_flo = 1.23

num_new = num_int + num_flo

print("datatype of num_int:",type(num_int))
print("datatype of num_flo:",type(num_flo))

print("Value of num_new:",num_new)
print("datatype of num_new:",type(num_new))
```

Type Conversion/Casting

❑ **Explicit Type Conversion**

In Explicit Type Conversion, users convert the data type of an object to required data type. We use the predefined functions like int(), float(), str(), etc to perform explicit type conversion.

```
num_int = 123
num_str = "456"

print("Data type of num_int:",type(num_int))
print("Data type of num_str:",type(num_str))

print(num_int+num_str)
```

```
num_int = 123
num_str = "456"

print("Data type of num_int:",type(num_int))
print("Data type of num_str before Type
Casting:",type(num_str))

num_str = int(num_str)
print("Data type of num_str after Type Casting:",type(num_str))

num_sum = num_int + num_str

print("Sum of num_int and num_str:",num_sum)
print("Data type of the sum:",type(num_sum))
```

Numeric Data Types

Python Number Types: **int, float**

Int

In Python, integers are zero, positive or negative whole numbers without a fractional part

```
num=100
print(num)
```

Float

In Python, floating point numbers (float) are positive and negative real numbers with a fractional part denoted by the decimal symbol.

```
num=100.25
print(num)
```

Boolean Data Types

boolean

The boolean value can be of two types only i.e. either True or False. The output <class ‘bool’> indicates the variable is a boolean data type.

```
a = True
print(a)
print(type(a))

b = False
print(b)
print(type(b))
```

Swapping

```
a = 10
b = 20

temp = a
a = b
b = temp

print(a, b)
```

```
a = 10
b = 20

a = a+b
b = a - b
a = a -b

print(a, b)
```

```
a = 10
b = 20

a, b = b, a

print(a, b)
```

Strings

Strings are List like many other popular programming languages. Python does not have a character data type, a single character is simply a string with a length of 1. Square brackets can be used to access elements of the string.

0	1	2	3	4	5	6	7	8	9	10
H	e	l	l	o		W	o	r	l	d
-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

```
a = "Hello World"
print(a)
print(a[0])
print(a[-1])
print(a[0:3])
print(a[0:])
print(a[1:])
print(a[:4])
print(a[0:-1])
```

Language Component & Loop

By Aksadur Rahman

aksadur@yahoo.com

Agenda

Arithmetic Operations
Operator Precedence
Math Functions
Indentation
If Statements
Logical Operators
Comparison/Relational/Conditional Operators
Assignment Operators
Ternary Operators
While Loops
Break & Continue Statement
Sum of n Numbers Program
For Loops
For-While Comparison
For with Range Function
Nested Loops

Arithmetic Operations

Python Arithmetic Operators

Arithmetic operators are used with numeric values to perform common mathematical operations:

Operator	Name	Example
+	Addition	x + y
-	Subtraction	x - y
*	Multiplication	x * y
/	Division	x / y
%	Modulus	x % y
**	Exponentiation	x ** y
//	Floor division	x // y

Arithmetic Operations

Example

```
val1 = 3
val2 = 2

print(val1 + val2)
print(val1 - val2)
print(val1 * val2)
print(val1 / val2)
print(val1 // val2)
print(val1 % val2)
print(val1 ** val2)
```

Operator Precedence

Operators	Meaning
()	Parentheses
**	Exponent
*, /, //, %	Multiplication, Division, Floor division, Modulus
+, -	Addition, Subtraction

```
print(10+3*2**2+45)
```

Math Functions/Module

Python math Functions

Python Math functions is one of the most used functions in Python Programming. In python there are different built-in math functions. Beside there is also a math module in python.

```
x=2.9
print(round(x))
print(abs(-2.9))
```

Python math Module

Python has a built-in module that you can use for mathematical tasks. The math module has a set of methods and constants.

```
import math

x=2.9
print(math.ceil(x))
print(math.floor(x))
```

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

Python Calendar

```
import calendar

year = 2024

print(calendar.calendar(year))
```

Python Indentation

Python Indentation

Indentation refers to the spaces at the beginning of a code line.

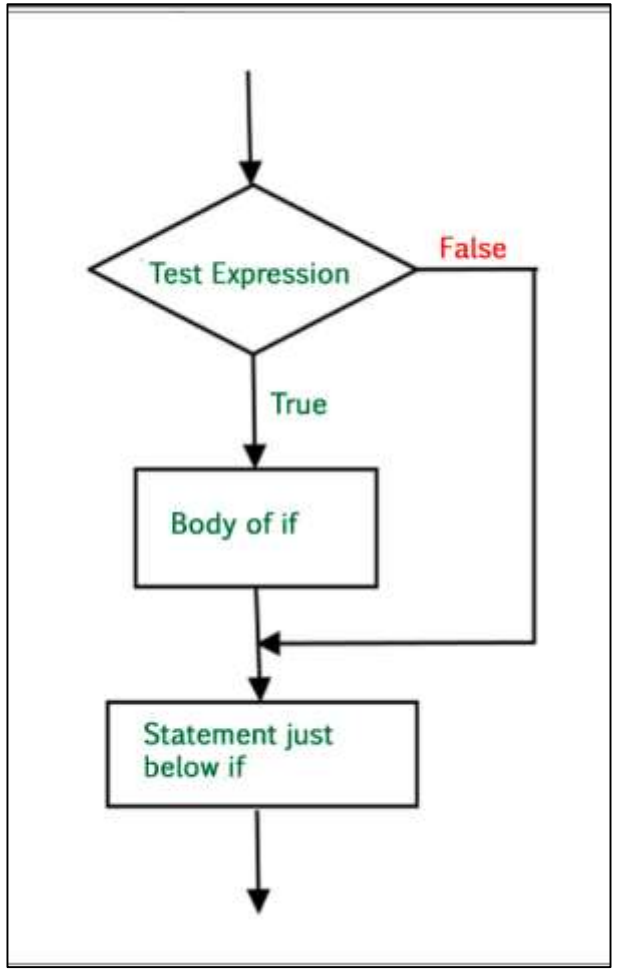
```
if 5 > 2:
    print("Five is greater than two!")
```

If Statements

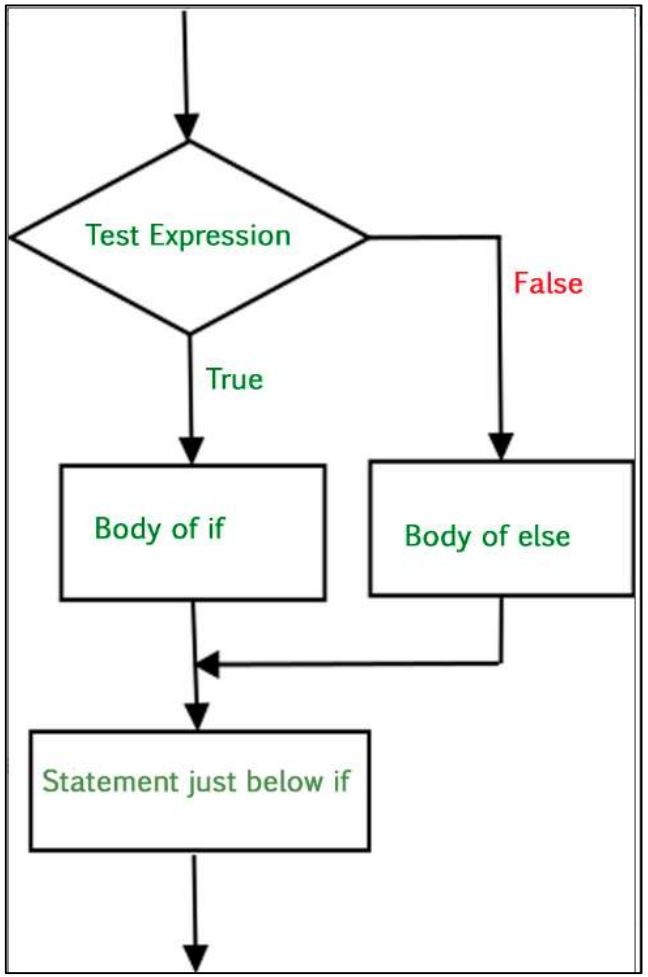
if statement

if statement is the most simple decision-making statement. It is used to decide whether a certain statement or block of statements will be executed or not i.e if a certain condition is true then a block of statement is executed otherwise not.

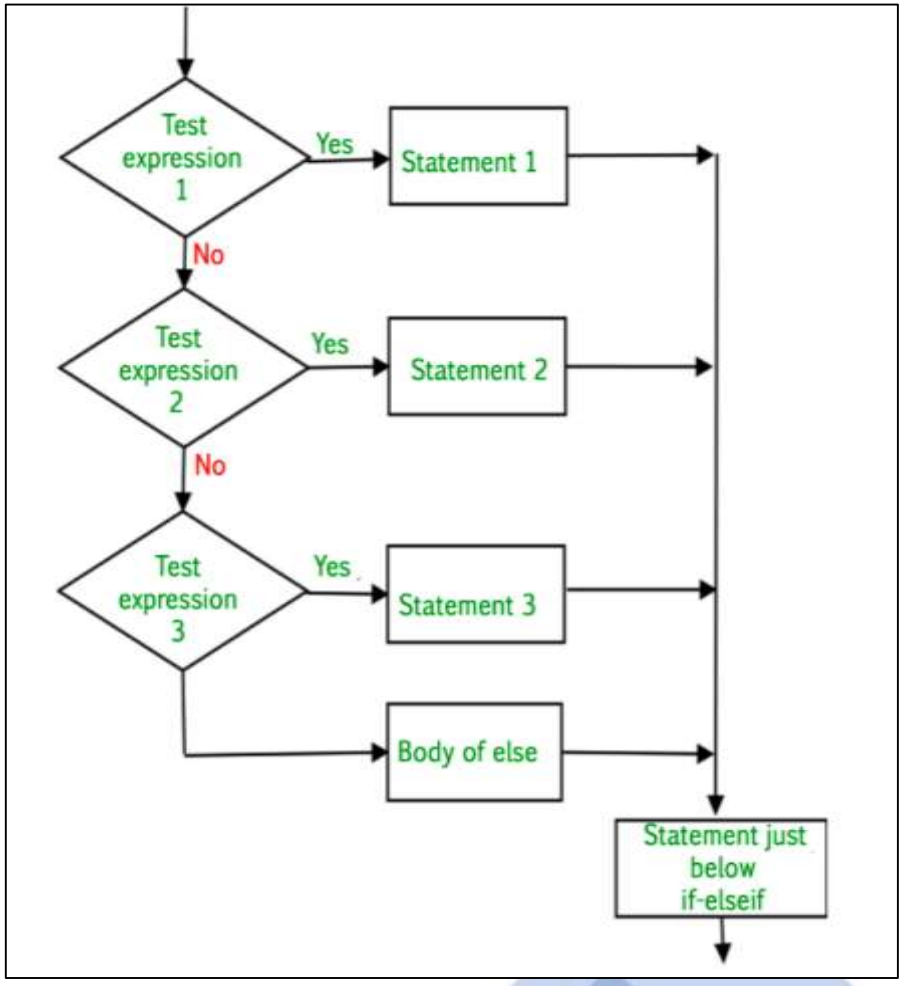
if



If……else



If……elif……else



If Statements

Example

if

```
i = 10

if i > 15:
    print("10 is less than 15")
print("I am Not in if")
```

If……else

```
i = 20
if i < 15:
    print("i is smaller than 15")
    print("i'm in if Block")
else:
    print("i is greater than 15")
    print("i'm in else Block")
print("i'm not in if and not in else Block")
```

If……elif……else

```
i = 20
if i == 10:
    print("i is 10")
elif i == 15:
    print("i is 15")
elif i == 20:
    print("i is 20")
else:
    print("i is not present")
```

Match Case

Example

```
num = 3
match num:
    # pattern 1
    case 1:
        print("One")
    # pattern 2
    case 2:
        print("Two")
    # pattern 3
    case 3:
        print("Three")
    # default pattern
    case _:
        print("Number not between 1 and 3")
```

Comparison/Relational/Conditional Operators

>	Greater than: True if the left operand is greater than the right	x > y
<	Less than: True if the left operand is less than the right	x < y
==	Equal to: True if both operands are equal	x == y
!=	Not equal to – True if operands are not equal	x != y
>=	Greater than or equal to: True if left operand is greater than or equal to the right	x >= y
<=	Less than or equal to: True if left operand is less than or equal to the right	x <= y

Comparison/Relational/Conditional Operators

Example:

```
a = 9
b = 5

print(a > b)
print(a < b)
print(a == b)
print(a != b)
print(a >= b)
print(a <= b)
```

Logical Operators

OPERATOR	DESCRIPTION	SYNTAX
and	Logical AND: True if both the operands are true	x and y
or	Logical OR: True if either of the operands is true	x or y
not	Logical NOT: True if operand is false	not x

and

```
a = 10
b = 10
c = -10

if a > 0 and b > 0:
    print("The numbers are greater than 0")

if a > 0 and b > 0 and c > 0:
    print("The numbers are greater than 0")
else:
    print("Atleast one number is not greater than 0")
```

Logical Operators

or

```
a = 10
b = -10
c = 0

if a > 0 or b > 0:
    print("Either of the number is greater than 0")
else:
    print("No number is greater than 0")

if b > 0 or c > 0:
    print("Either of the number is greater than 0")
else:
    print("No number is greater than 0")
```

not

```
a = 10

if not (a%3 == 0 or a%5 == 0):
    print("10 is not divisible by either 3 or 5")
else:
    print("10 is divisible by either 3 or 5")
```

Assignment Operators

Python Assignment Operators

Assignment operators are used to assign values to variables:

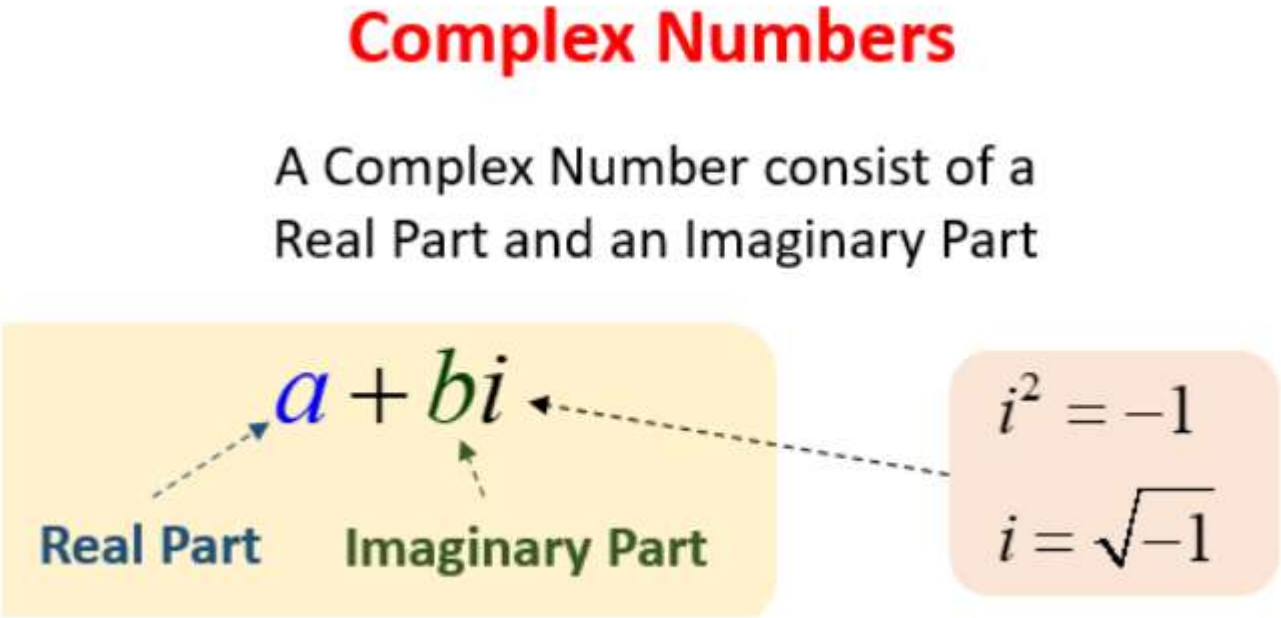
Operator	Example	Same As
=	x = 5	x = 5
+=	x += 3	x = x + 3
-=	x -= 3	x = x - 3
*=	x *= 3	x = x * 3
/=	x /= 3	x = x / 3
%=	x %= 3	x = x % 3
//=	x //= 3	x = x // 3
**=	x **= 3	x = x ** 3

Numeric Data Types

Python Number Types: **int, float, complex**

Complex

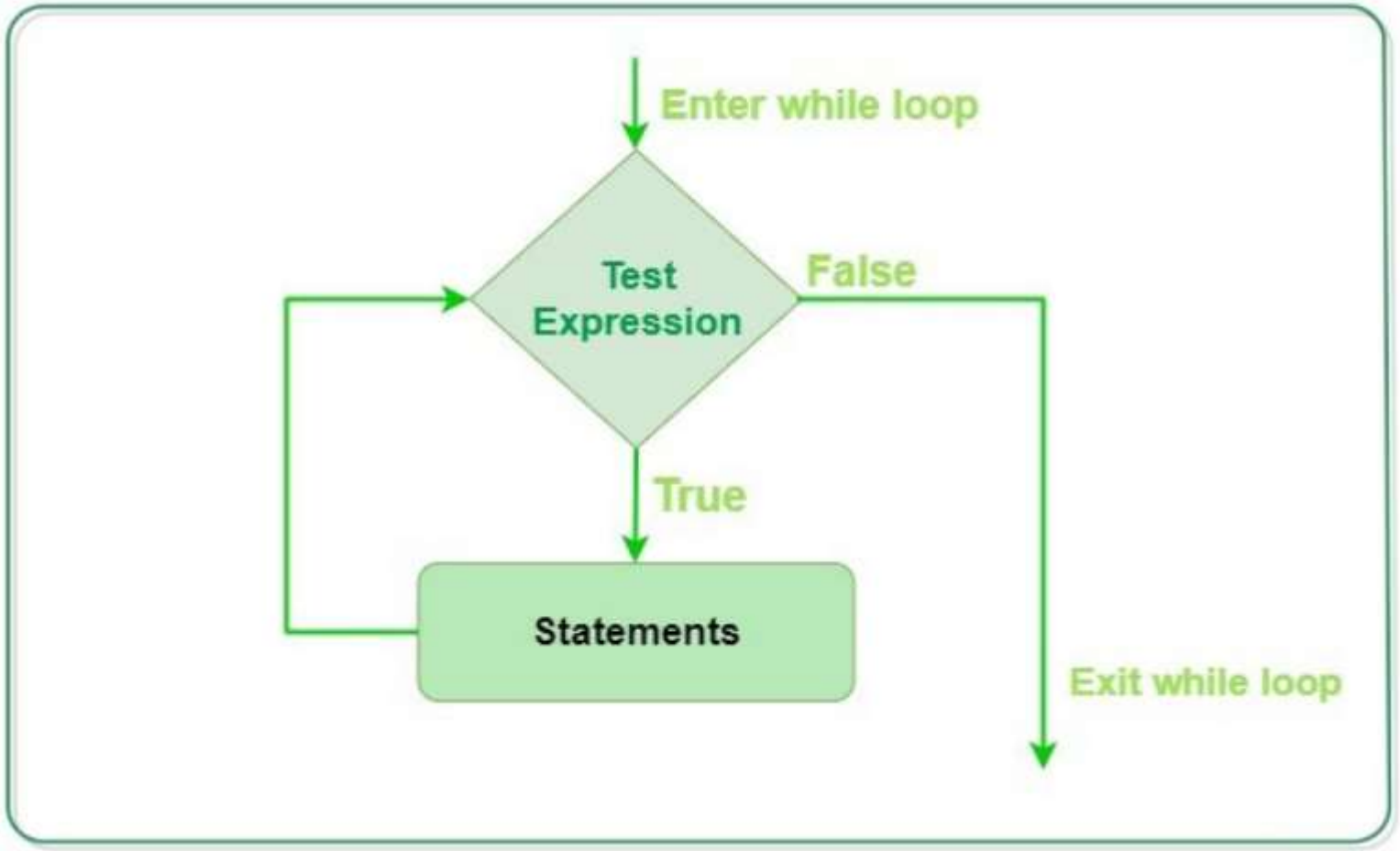
A complex number is a number with real and imaginary components. For example, 5 + 6j is a complex number where 5 is the real component and 6 multiplied by j is an imaginary component.
Complex data types is used while developing scientific applications where complex mathematical operation is required.




```
a=6+7j
print(a)
print(a.real)
print(a.imag)
print(type(a))

b=5+5j
print(a+b)
```

While Loops

A while loop statement in Python programming language repeatedly executes a target statement as long as a given condition is true.




```
count = 0
while count < 9:
  print('The count is:', count)
  count = count + 1

print("Good bye!")
```

Sum of n Numbers Program

```
n = int(input("Enter the n Number:"))
sum = 0
i = 1

while i <= n:
    sum = sum + i
    i = i +1

print(sum)
```

Break & Continue Statement

Break

```
i = 1
while i < 6:
  print(i)
  if i == 3:
    break
  i += 1
```

Continue

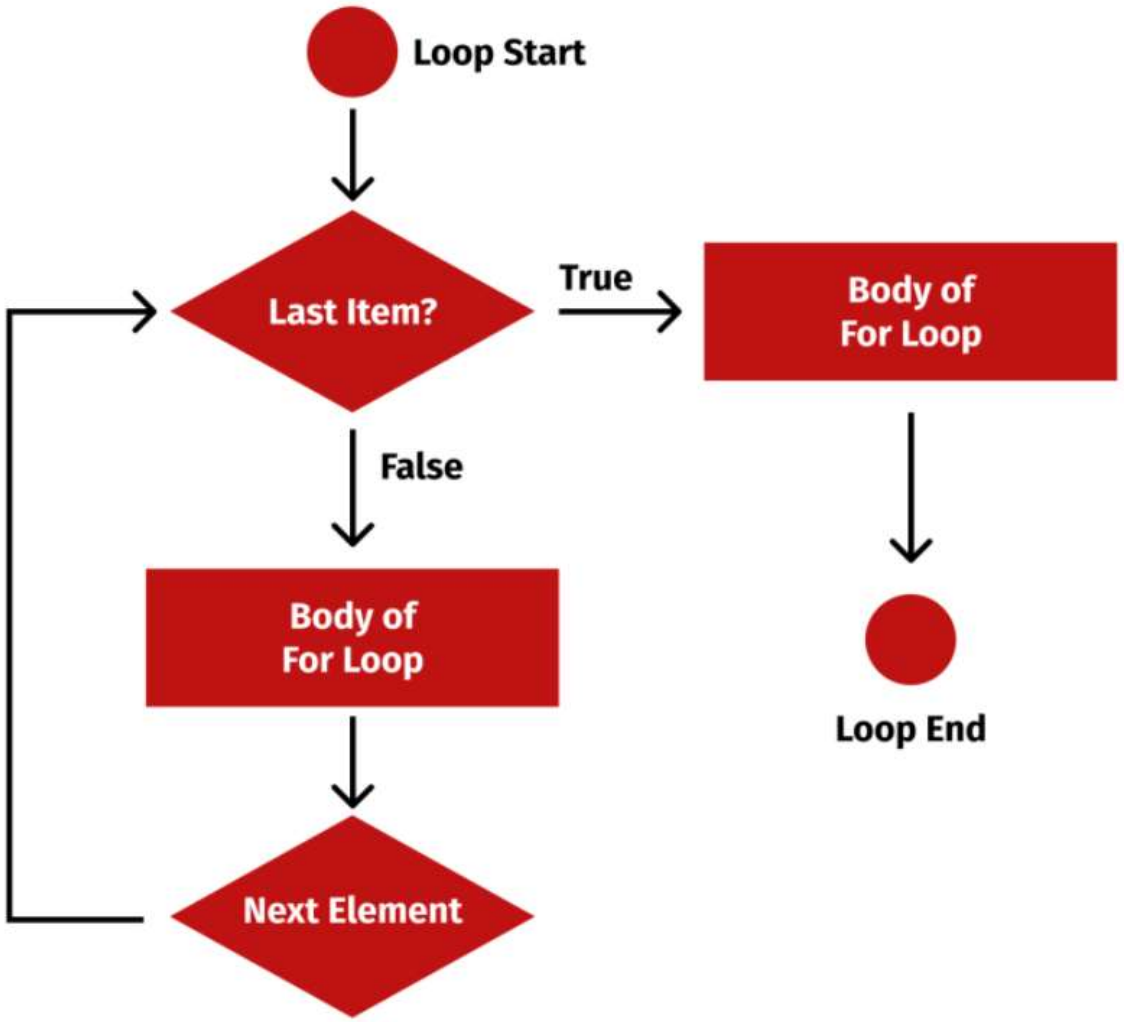
```
i = 0
while i < 6:
  i += 1
  if i == 3:
    continue
  print(i)
```

```
sum = 0

while True:
    num = input("Enter a Number: ")
    if num == "quit":
        break

    try:
        num = int(num)
    except:
        print("Enter a valid number please.")
        continue
    sum = sum + num
    print(sum)
```

For Loops

For Loops

A for loop is used for iterating over a sequence (that is either a list, a tuple, a dictionary, a set, or a string).



For Loops

```
#Looping Through a String

for x in "banana":
  print(x)
```

```
#Looping Through a list

fruits = ["apple", "banana", "cherry"]
for x in fruits:
  print(x)
```

```
#The break Statement
#Exit the loop when x is "banana"

fruits = ["apple", "banana", "cherry"]
for x in fruits:
  print(x)
  if x == "banana":
    break
```

```
#The continue Statement
#Do not print banana:

fruits = ["apple", "banana", "cherry"]
for x in fruits:
  if x == "banana":
    continue
  print(x)
```

For-While Comparison

while

```
num = [10, 20, 30, 40, 50]
index = 0
n = len(num)
while index < n:
    print(num[index])
    index = index+1
```

for

```
num = [10, 20, 30, 40, 50]
for x in num:
    print(x)
```

For with Range Function

The range() Function

To loop through a set of code a specified number of times, we can use the range() function,

```
#Using the range() function:

for x in range(6):
  print(x)
```

```
#Using the start parameter:

for x in range(2, 6):
  print(x)
```

```
#Increment the sequence with 3 (default is 1):

for x in range(2, 30, 3):
  print(x)
```

For with Enumerate

```
num = [30, 10, 70, 12]

for i, x in enumerate(num):
    print(i, x)
```

Nested Loops

```
adj = ["red", "big", "tasty"]
fruits = ["apple", "banana", "cherry"]

for x in adj:
  for y in fruits:
    print(x, y)
```