



**Marwadi**  
University

01CE1705-Programming with Python

# Unit-1 Basic of Python

Prof. Chetan Chudasama  
Computer Engineering Department



# Outline

- Introduction
- Installation
- Features
- Variable
- Python Interpreter and its working
- Data Types
- Operators
- Expressions
- Comments

# Introduction

- Python is General purpose(everything), high level programming language(machine independent).
- It was created by Guido van Rossum and released in 1991.
- It can be Used for

Console application(text-based interaction)

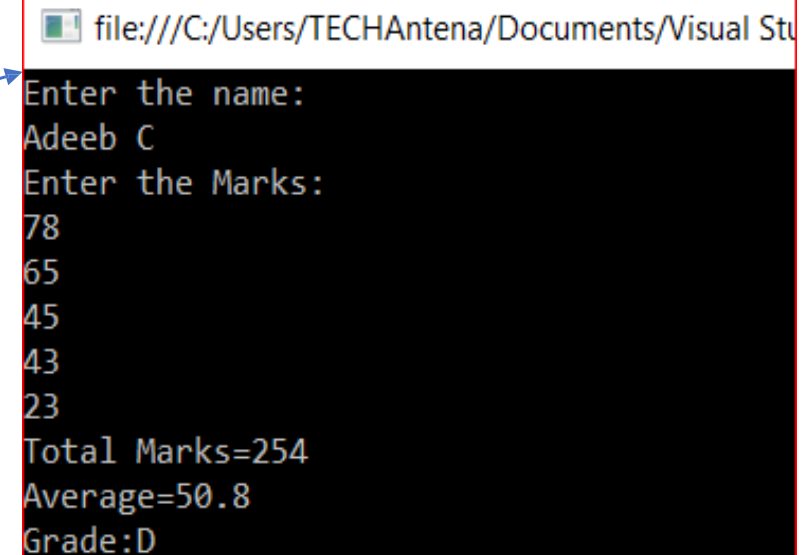
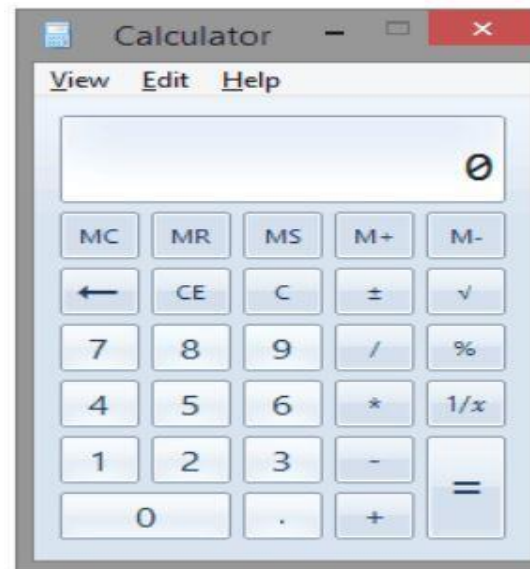
Desktop application(Calculator)

Web Site

Mobile application

Machine learning

IOT applications

A screenshot of a Python console application window. The title bar shows the file path: "file:///C:/Users/TECHAntena/Documents/Visual Stu". The console output is as follows:

```
Enter the name:  
Adeeb C  
Enter the Marks:  
78  
65  
45  
43  
23  
Total Marks=254  
Average=50.8  
Grade:D
```

# Popular apps developed in Python

 YouTube



Google

Quora

bitly

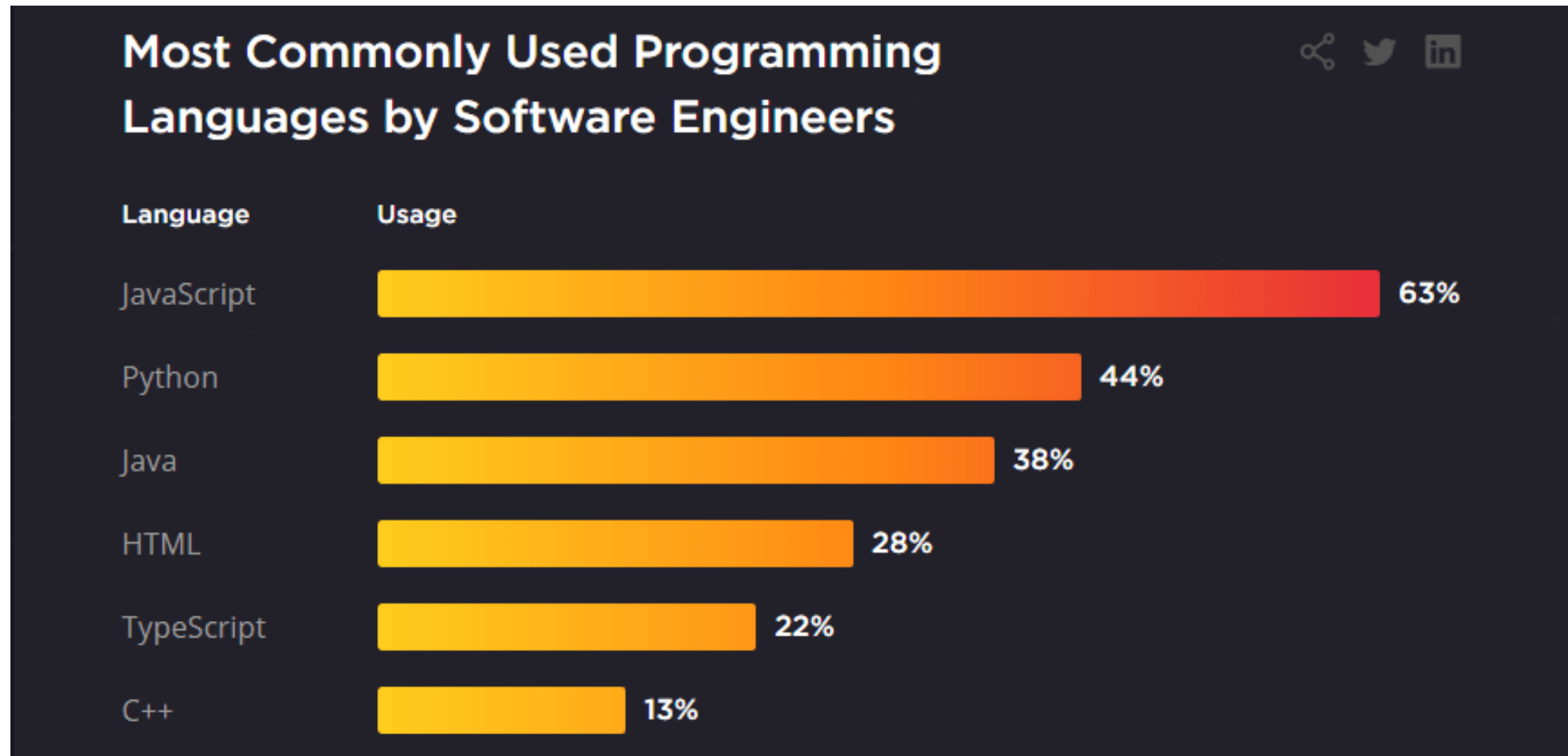
reddit

Instagram

Pinterest

DISQUS

# Most popular languages



- Python is Case sensitive.

```
a=10
```

```
print(A) → Error name 'A' is not defined
```

- It is an Object-Oriented Language.
- There are two major version of Python, Python 2 and Python 3.
  - 1) Python 2.0, released in 2000,
    - ❖ Introduced features like **list comprehensions** and a **garbage collection system**
  - 2) Python 3.0 released in 2008 and current version of python is 3.11.4 (as of June-2023).
    - ❖ The Python 2 language was officially discontinued in 2020

# Installation



- For Windows & Mac:
  - To install python in windows you need to download installable file from <https://www.python.org/downloads>
  - After downloading the installable file, you need to execute the file.
- For Linux
  - For ubuntu 16.10 or newer
    - `sudo apt-get install python`
- To verify installation
  - Windows:
    - `python --version`
  - Linux:
    - `python3 --version`(linux might have python2 already installed, you can check python2 using `python --version`)
- Alternatively we can use anaconda distribution for python installation
  - <http://anaconda.com/downloads>
  - Anaconda comes with many useful libraries



# Installing Python on Windows



download python



Tools

About 70,40,00,000 results (0.32 seconds)

<https://www.python.org> › downloads

## Download Python - Python.org

Download the latest version of Python. **Download Python 3.10.5**. Looking for Python with a different OS? Python for Windows, Linux/UNIX, macOS, Other.

[Python Releases for Windows](#) · [Python 3.9.0](#) · [Python 3.10.0](#) · [Python 3.10.4](#)

### People also ask

Can I download Python for free?



How do I install Python on my PC?



How do I download Python?



Where can I download Python for PC?



Feedback

<https://www.python.org> › downloads › windows

## Python Releases for Windows

The official home of the Python Programming Language.



Python

PSF

Docs

PyPI

Jobs

Community



Donate



Search

GO

Socialize

About

Downloads

Documentation

Community

Success Stories

News

Events

## Download

Download Python

Looking for Python

[Linux/UNIX, macOS](#)

Want to help test d

[Docker images](#)

Looking for Python

All releases

Source code

Windows

macOS

Other Platforms

License

Alternative Implementations

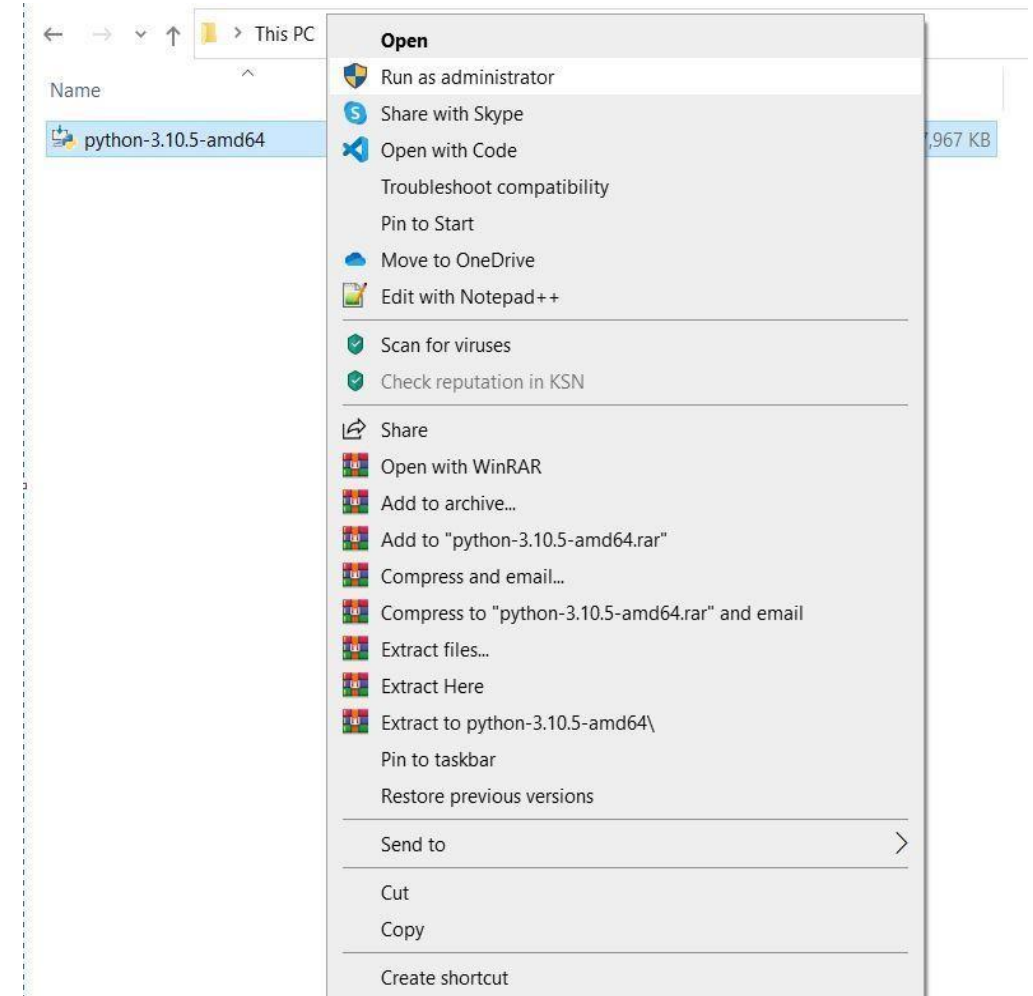
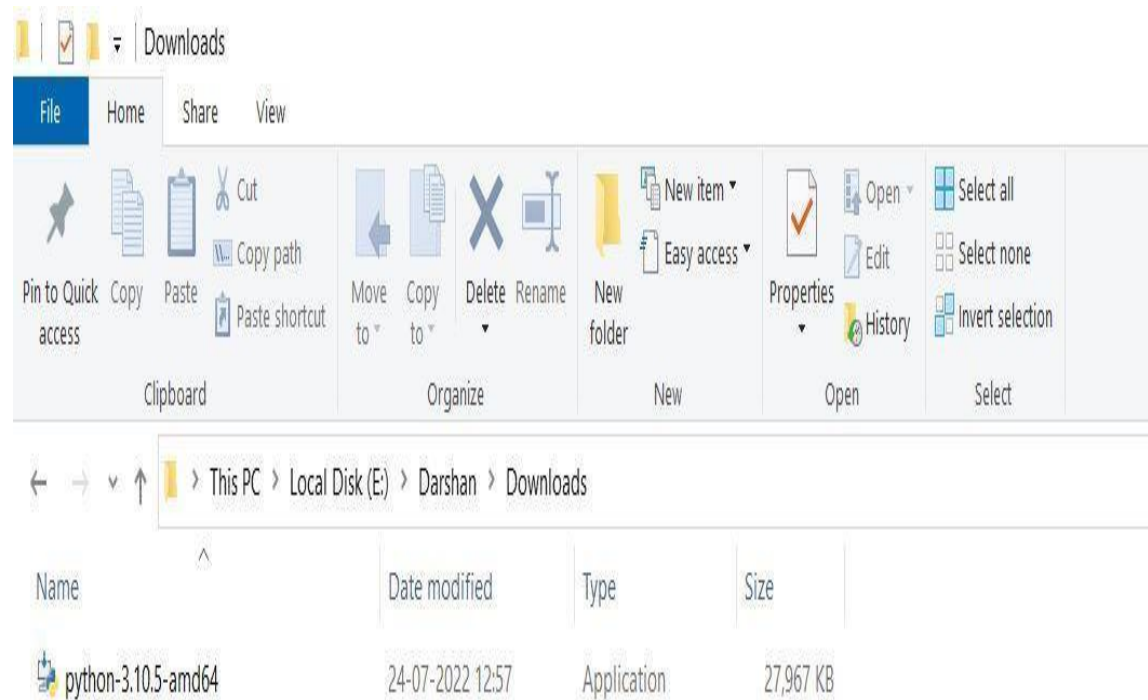
### Download for Windows

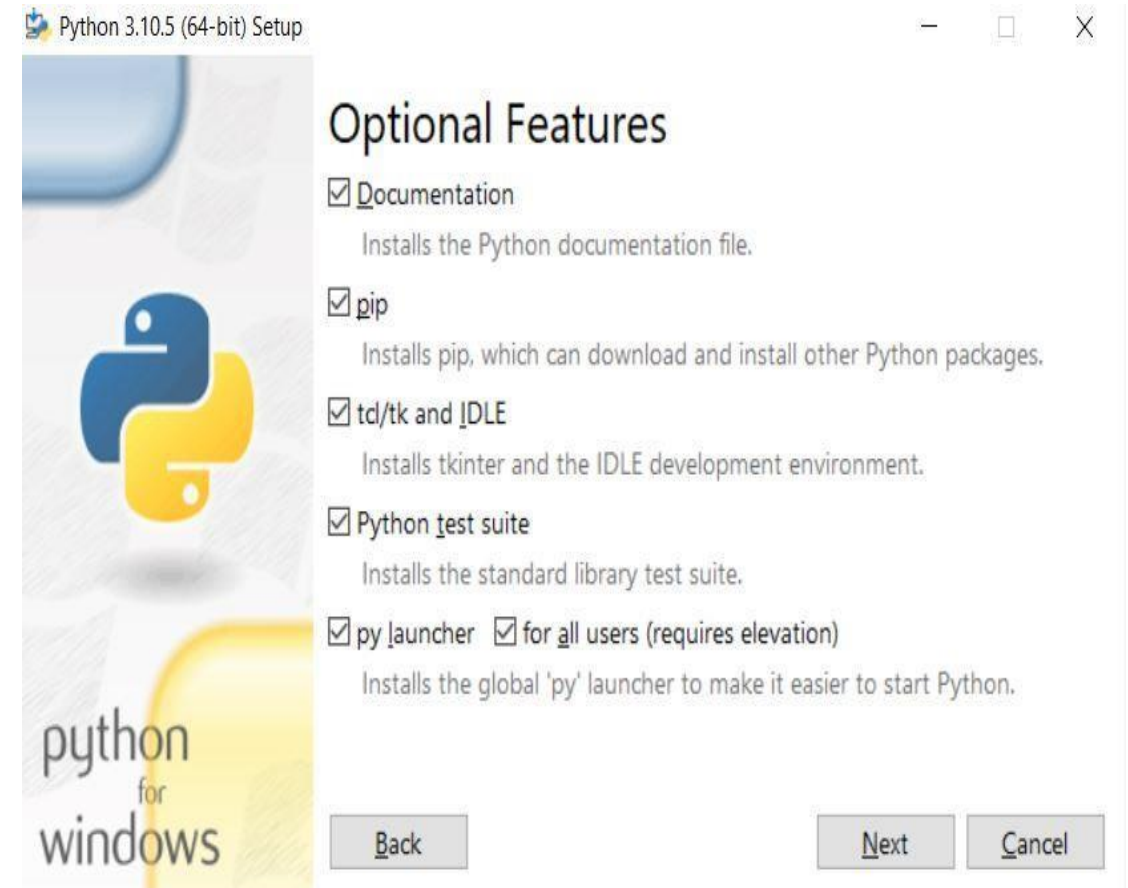
Python 3.10.5

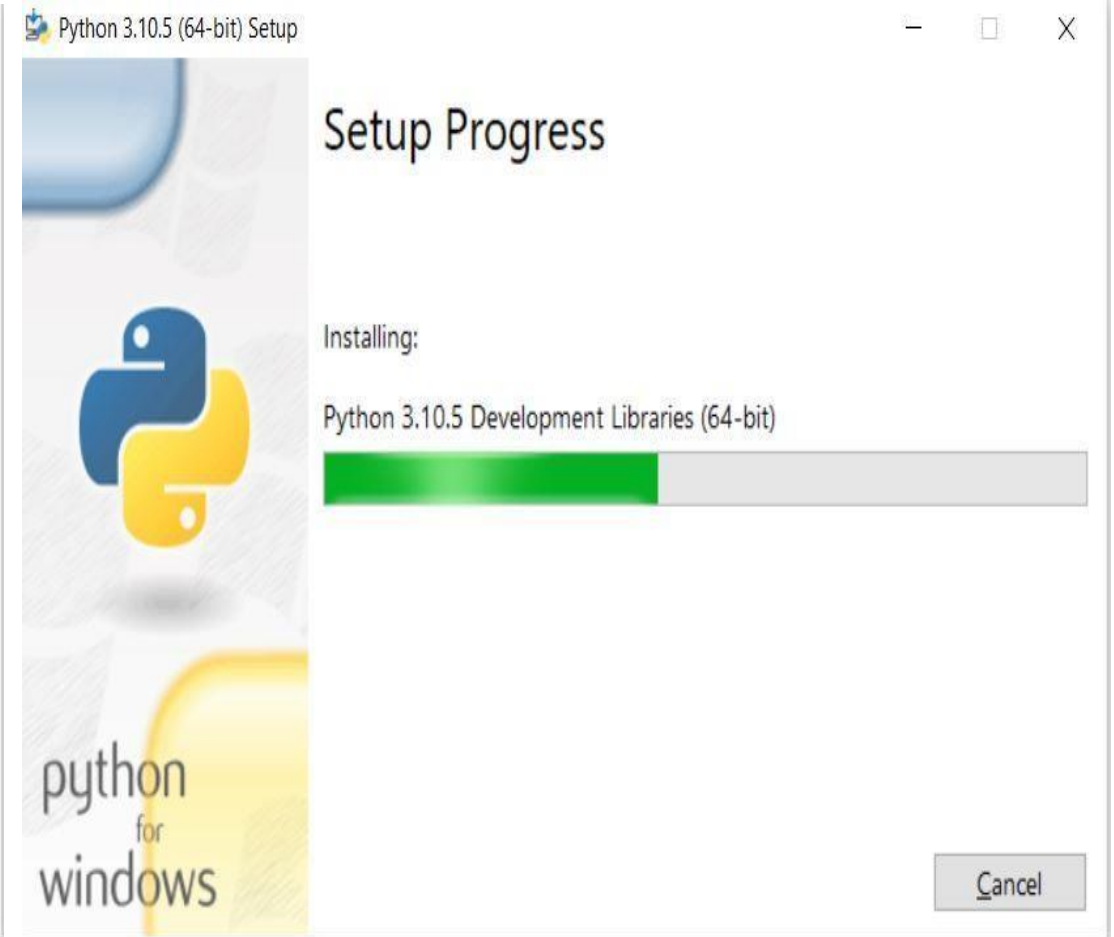
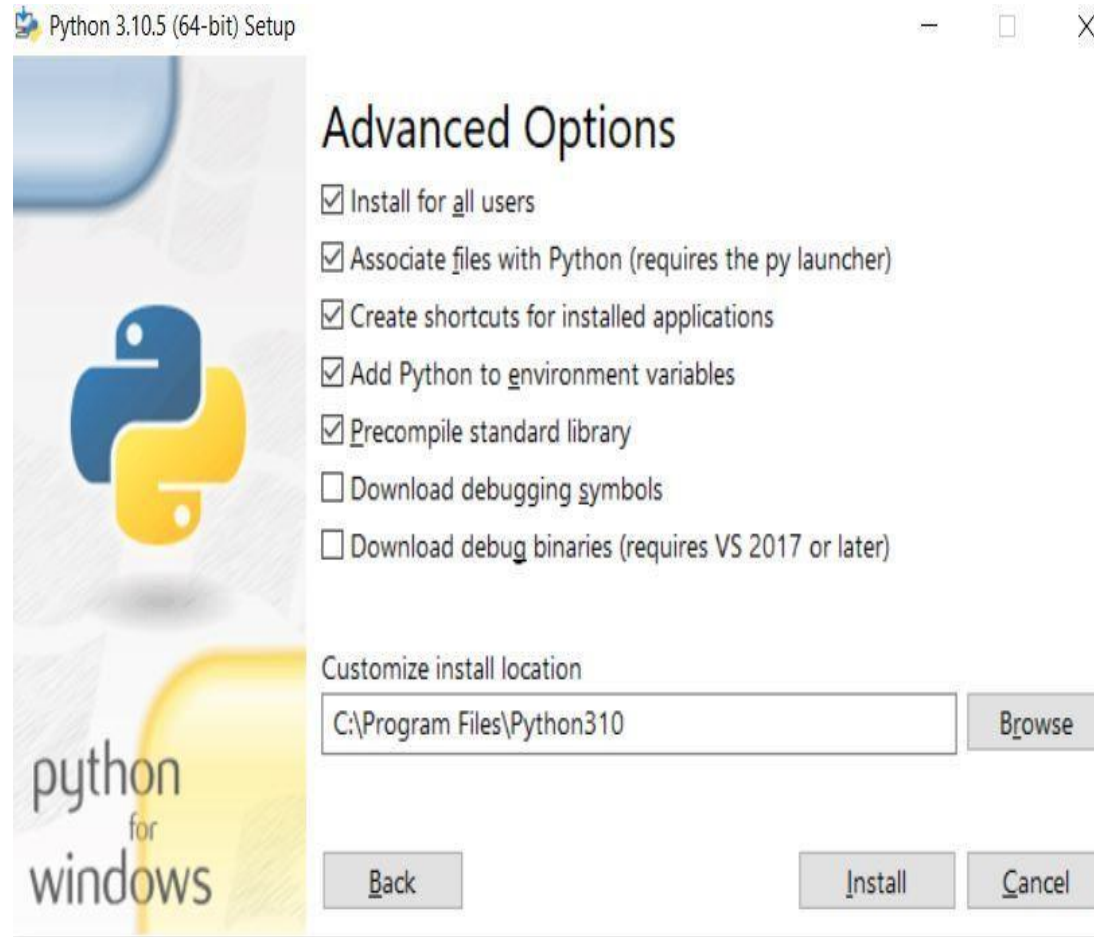
**Note that Python 3.9+ *cannot* be used on Windows 7 or earlier.**

Not the OS you are looking for? Python can be used on many operating systems and environments.

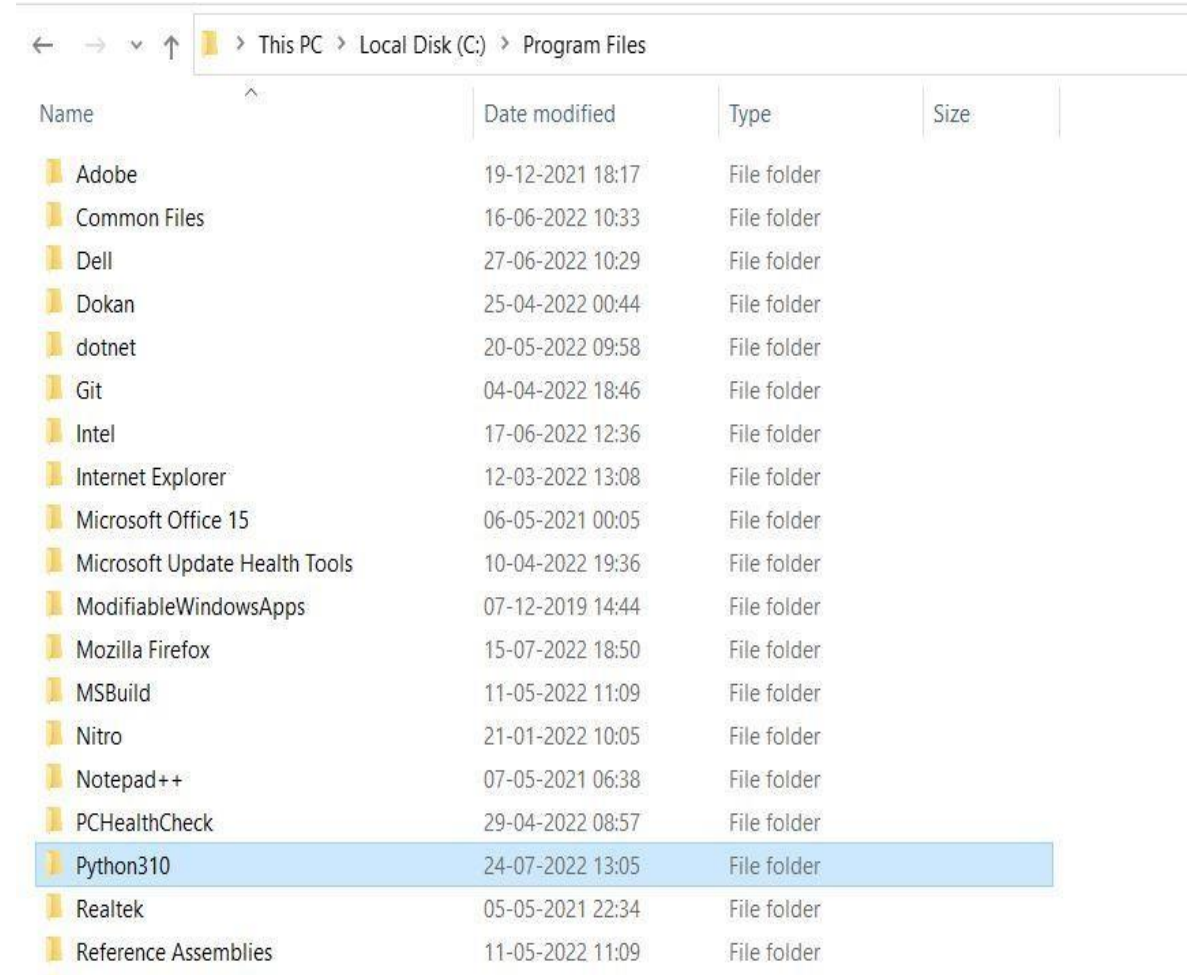
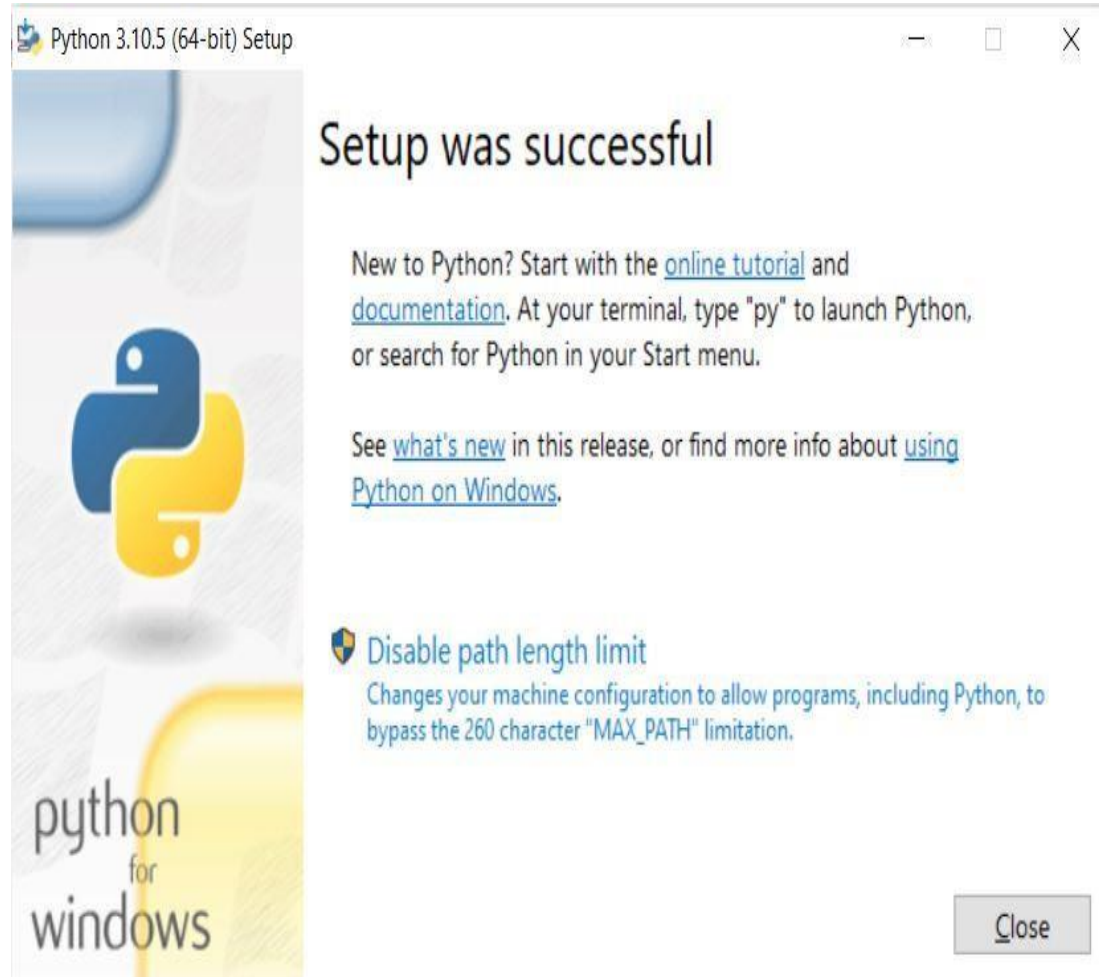
[View the full list of downloads.](#)











← → ↕ ↑ > This PC > Local Disk (C:) > Program Files

Name	Date modified	Type	Size
Adobe	19-12-2021 18:17	File folder	
Common Files	16-06-2022 10:33	File folder	
Dell	27-06-2022 10:29	File folder	
Dokan	25-04-2022 00:44	File folder	
dotnet	20-05-2022 09:58	File folder	
Git	04-04-2022 18:46	File folder	
Intel	17-06-2022 12:36	File folder	
Internet Explorer	12-03-2022 13:08	File folder	
Microsoft Office 15	06-05-2021 00:05	File folder	
Microsoft Update Health Tools	10-04-2022 19:36	File folder	
ModifiableWindowsApps	07-12-2019 14:44	File folder	
Mozilla Firefox	15-07-2022 18:50	File folder	
MSBuild	11-05-2022 11:09	File folder	
Nitro	21-01-2022 10:05	File folder	
Notepad++	07-05-2021 06:38	File folder	
PCHealthCheck	29-04-2022 08:57	File folder	
Python310	24-07-2022 13:05	File folder	
Realtek	05-05-2021 22:34	File folder	
Reference Assemblies	11-05-2022 11:09	File folder	

Command Prompt

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

```
C:\Users\Virat Kohli>python --version  
Python 3.10.5
```

```
C:\Users\Virat Kohli>pip --version  
pip 22.0.4 from C:\Program Files\Python310\lib\site-packages\pip (python 3.10)
```

```
C:\Users\Virat Kohli>
```

## Installing Python using Anaconda Distribution

- Anaconda is a package that contains different tools.
- This tools deals with Machine Learning, Artificial Intelligence, data science.
- Anaconda provide different tools such as  
Spyder  
Jupyter  
VS code
- Jupyter and Spyder is used in IT Companies.
- Jupyter is open in Web browser.





anaconda python



Tools

About 3,47,00,000 results (0.50 seconds)

<https://www.anaconda.com> › products › distribution

## Anaconda Distribution

**Anaconda** Distribution equips individuals to easily search and install thousands of **Python**/R packages and access a vast library of community content and support.

[Open Source](#) · [Pricing](#) · [Anaconda Professional](#) · [Anaconda FAQ](#)

<https://www.anaconda.com>

## Anaconda | The World's Most Popular Data Science Platform

**Anaconda** is the birthplace of **Python** data science. We are a movement of data scientists, data-driven enterprises, and open source communities.

<https://anaconda.org> › anaconda › python

## Python :: Anaconda.org

**Python** is a widely used high-level, general-purpose, interpreted, dynamic programming language. Its design philosophy emphasizes code readability, and its ...

<https://anaconda.org>

## :: Anaconda.org

Expedite your data science journey with easy access to training materials, how-to videos, and expert insights on **Anaconda** Nucleus, all free for a limited ...



[Products](#) ▾

[Pricing](#)

[Solutions](#) ▾

[Resources](#) ▾

[Partners](#) ▾

[Blog](#)

[Company](#) ▾

[Contact Sales](#)

Individual Edition is now

# ANACONDA DISTRIBUTION

The world's most popular open-source Python distribution platform

## Anaconda Distribution

[Download](#) 

For Windows

Python 3.9 • 64-Bit Graphical Installer • 594 MB

Get Additional Installers



← → ↑ ↓ This PC > Local Disk (C:) > Users > DELL > Downloads

Name	Date modified	Type	Size
▼ Today (1)			
Anaconda3-2022.05-Windows-x86_64	31-07-2022 09:51 AM	Application	6,08,137 KB
▼ Yesterday (5)			
roboto	30-07-2022 11:34 PM	WinRAR ZIP archive	836 KB
OfficeSetup	30-07-2022 11:02 PM	Application	8,228 KB
SupportAssistantInstaller	30-07-2022 09:21 PM	Application	666 KB
startup	30-07-2022 09:08 PM	Application	2,693 KB
roboto	30-07-2022 11:35 PM	File folder	

Quick access

- Desktop
- Downloads
- Documents
- Pictures
- imp files
- New Volume (E:)
- Python
- Python\_Images
- OneDrive - Personal
- This PC
- Network

← → ↑ ↓ This PC > Local Disk (C:) > Users > DELL > Downloads

Search Downloads

Name	Date modified	Type	Size
▼ Today (1)			
Anaconda3-2022.05-Windows-x86_64	31-07-2022 09:51 AM	Application	6,08,137 KB
▼ Yesterday (5)			
roboto	30-07-2022 11:34 PM	WinRAR ZIP archive	836 KB
OfficeSetup	30-07-2022 11:02 PM	Application	8,228 KB
SupportAssistantInstaller	30-07-2022 09:21 PM	Application	666 KB
startup	30-07-2022 09:08 PM	Application	2,693 KB
roboto	30-07-2022 11:35 PM	File folder	

Quick access

- Desktop
- Downloads
- Documents
- Pictures
- imp files
- New Volume (E:)
- Python
- Python\_Images
- OneDrive - Personal
- This PC
- Network

Open

- Run as administrator
- Troubleshoot compatibility
- Pin to Start
- Move to OneDrive
- Scan for viruses
- Check reputation in KSN
- Share
- Give access to
- Add to archive...
- Add to "Anaconda3-2022.05-Windows-x86\_64.rar"
- Compress and email...
- Compress to "Anaconda3-2022.05-Windows-x86\_64.rar" and email
- Pin to taskbar
- Restore previous versions
- Send to
- Cut
- Copy
- Create shortcut
- Delete
- Rename
- Properties

6 items 1 item selected 593 MB

09:53 AM 31-07-2022

Anaconda3 2022.05 (64-bit) Setup



## Welcome to Anaconda3 2022.05 (64-bit) Setup

Setup will guide you through the installation of Anaconda3 2022.05 (64-bit).

It is recommended that you close all other applications before starting Setup. This will make it possible to update relevant system files without having to reboot your computer.

Click Next to continue.

Next >

Cancel

Anaconda3 2022.05 (64-bit) Setup



### License Agreement

Please review the license terms before installing Anaconda3 2022.05 (64-bit).

Press Page Down to see the rest of the agreement.

=====

End User License Agreement - Anaconda Distribution

=====

Copyright 2015-2022, Anaconda, Inc.

All rights reserved under the 3-clause BSD License:

This End User License Agreement (the "Agreement") is a legal agreement between you and Anaconda, Inc. ("Anaconda") and governs your use of Anaconda Distribution (which was formerly known as Anaconda Individual Edition).

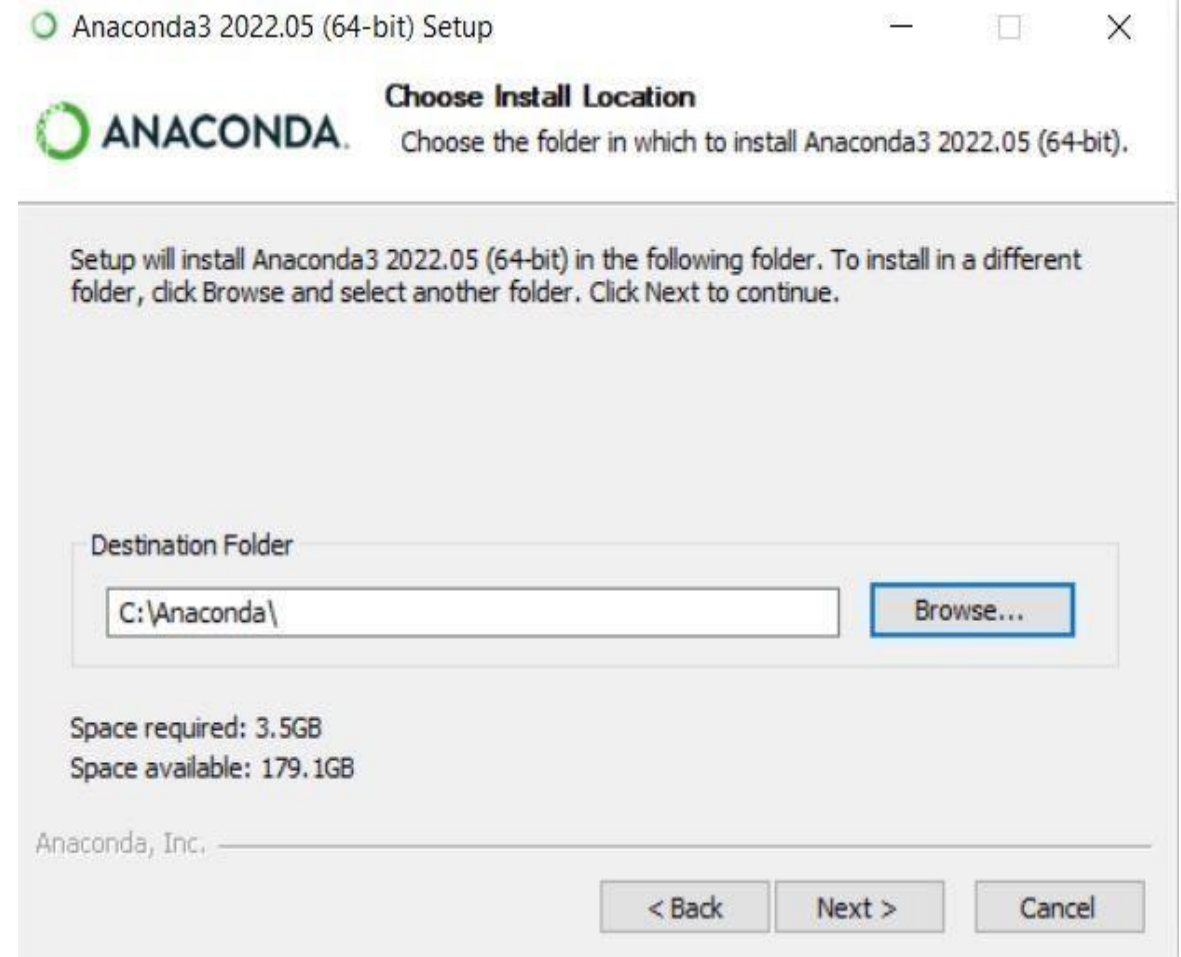
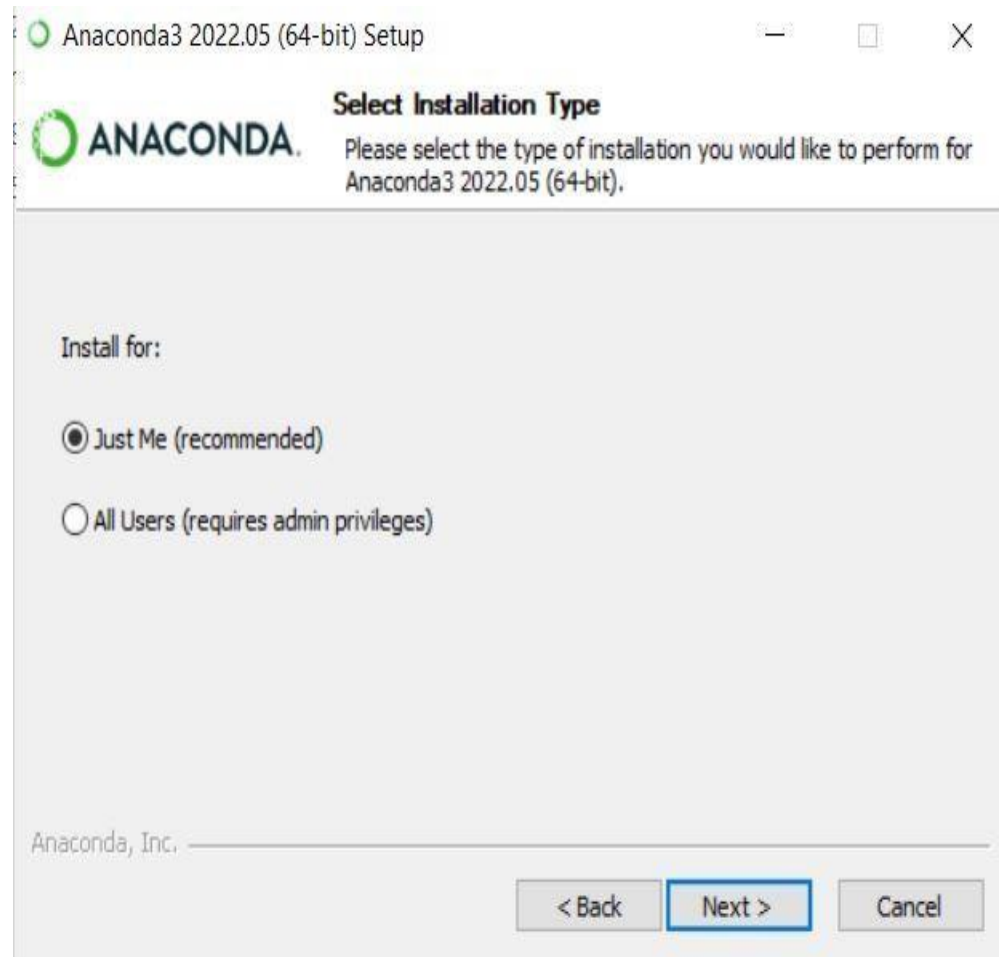
If you accept the terms of the agreement, click I Agree to continue. You must accept the agreement to install Anaconda3 2022.05 (64-bit).

Anaconda, Inc. \_\_\_\_\_

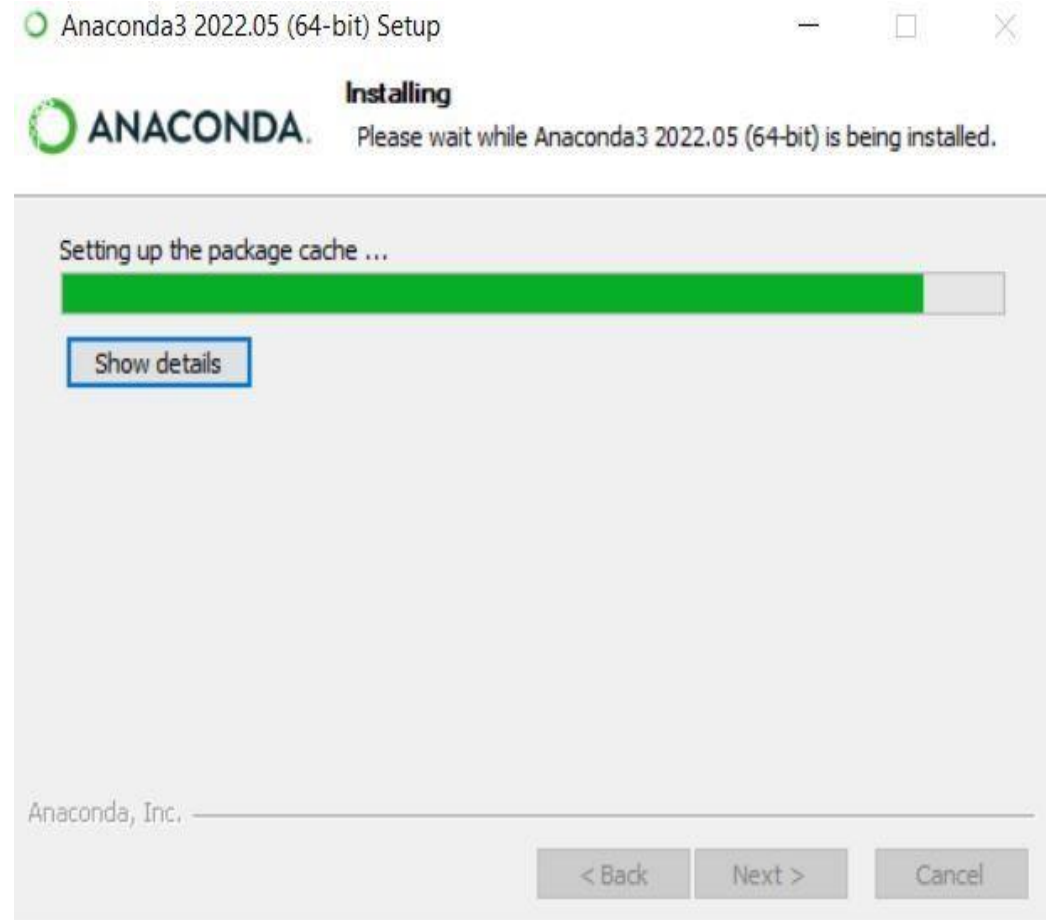
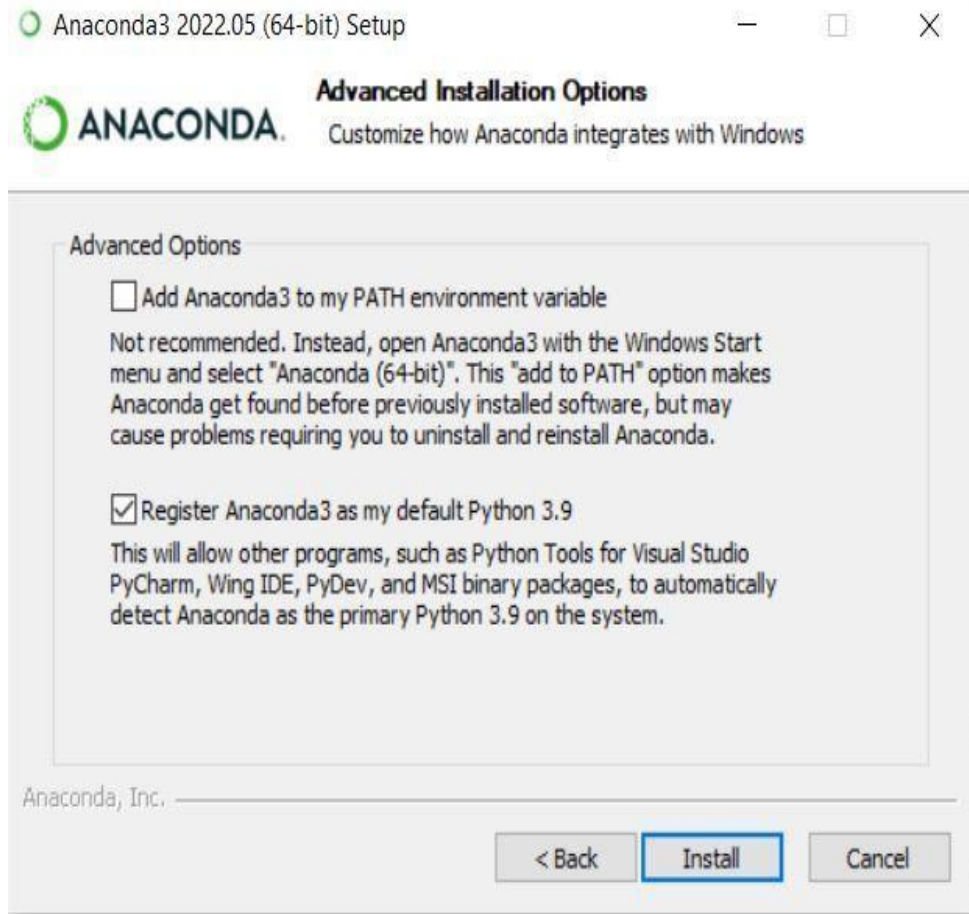
< Back

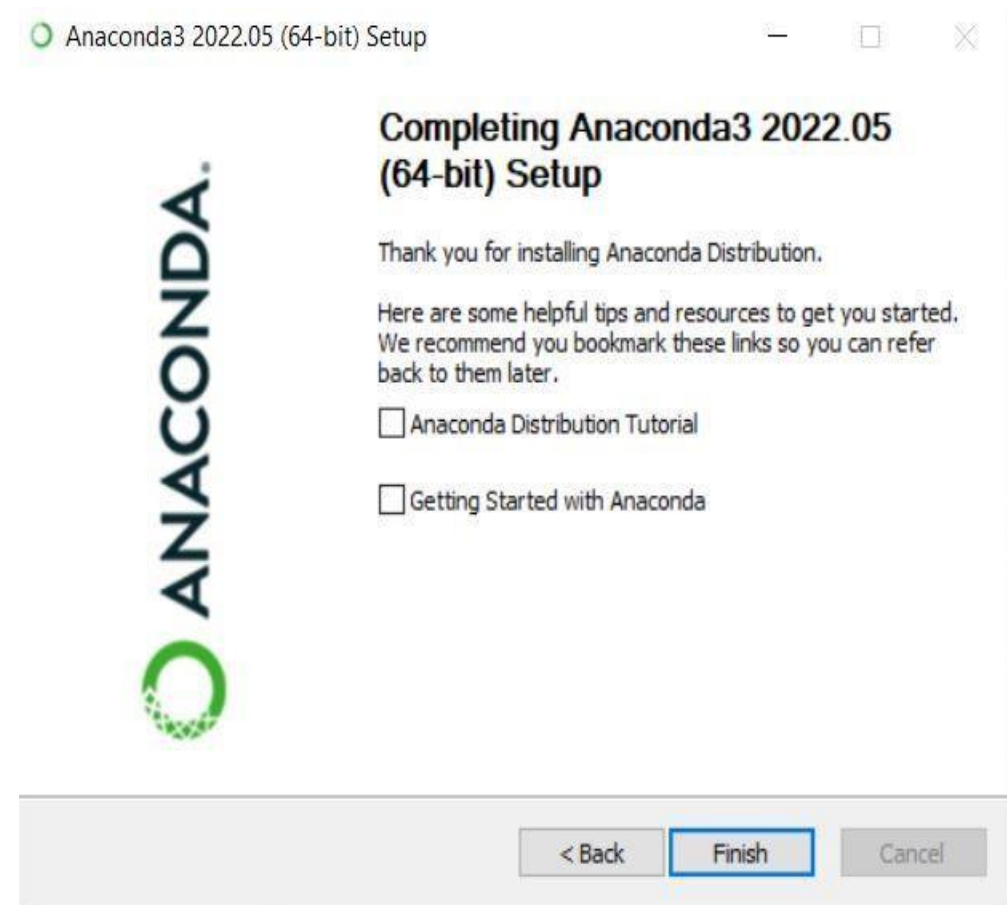
I Agree

Cancel









# Features of Python

## Easy to Read

- Python Code looks like simple English words.
- There is no use of semicolons or Curly-brackets.
- Indentation defines the code block.

**Note:-** Space at the beginning of code line.

- We can tell what the code is supposed to do simply by looking at it.

### C Language

```
#include<stdio.h>
void main()
{
    printf("Hello world");
}
```

### Python

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

### Indentation

```
i=1
while(i<=10):
    if(i%2==0):
        print(i)
    i+=1
```



## Dynamically Typed

- Python doesn't know the type of variable until we run the code.
- It automatically assigns the data type during execution.
- The programmer doesn't need to worry about declaring variables and their data types.

### C Language

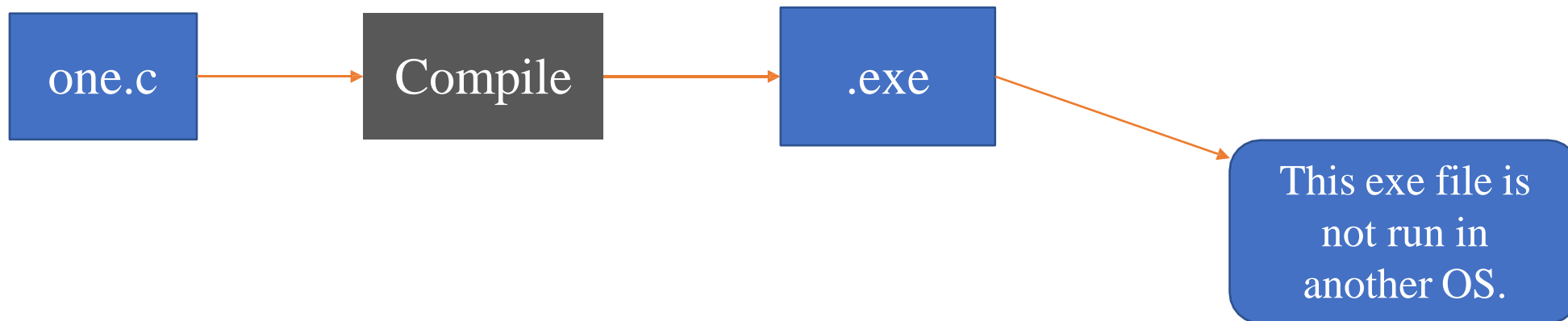
```
int a=10,b=20,c;  
c=a+b
```

### Python

```
a=10  
b=20  
c=a+b
```

## Portable

- In many languages like C/C++, you need to change your code to run the program on different platforms(different OS).
- That is not the same with Python. You only write once and run it anywhere.
- Python runs on all major operating systems like Microsoft Windows, Linux, and Mac OS.



## **GUI support**

- Graphical User Interface can be made using a module such as PyQt5, PyQt4, wxPython or Tkinter in python.
- PyQt5 is the most popular option for creating graphical apps with Python.

Module:-It is python file(functions, classes and variables)

## **High-Level Language**

- Python is a high-level language.
- When we write programs in python, we do not need to remember the system architecture, nor do we need to manage the memory.

## **Object-Oriented Language**

- Python supports object-oriented language and concepts of classes and objects.
- It supports inheritance, polymorphism, and encapsulation etc.

- The object-oriented procedure helps to programmer to write reusable code and develop applications in less code.

## **Database Support**

- Today almost every application we need to develop certainly requires a database.
- Some of the databases supported by standard python are MySQL, PostgreSQL, Microsoft SQL, Oracle, Informix, etc.

## **Interpreted**

- Python is an interpreted language which means that python directly executes the code line by line.
- In case of any error, it stops further execution and report back the error which has occurred.
- Python shows only one error even if the program has multiple errors. This makes debugging easier.

## **Large Standard Library**

- Python has an extensive standard library available for anyone to use.
- This means that programmers don't have to write their code for every single thing unlike other programming languages.
- There are libraries for image manipulation, databases and lot of other functionalities.
- Some machine learning libraries, such as Tensor flow, Pandas, Numpy, Keras, and Pytorch, etc. Django, flask, pyramids are the popular framework for Python web development.

## **Open-Source**

- Python is open-source and the community is always contributing to it to improve it.
- It is free and its source code is freely available to the public.

- We can download Python from the official Python Website.

## **Easy to Code**

- Anyone can learn to code in Python in few days.
- Mastering Python and all its advanced concepts, packages and modules might take some more time.
- Learning the basic Python syntax is very easy, as compared to other popular languages like C, C++, and Java.

**Note:-**

In Python Programming There are two modes in which we can run our code.

1.Interactive Mode

2.Script Mode

- Interactive mode is very convenient for writing very short lines of code.
- In the interactive mode as we enter a command and press enter, the very next step we get the output.

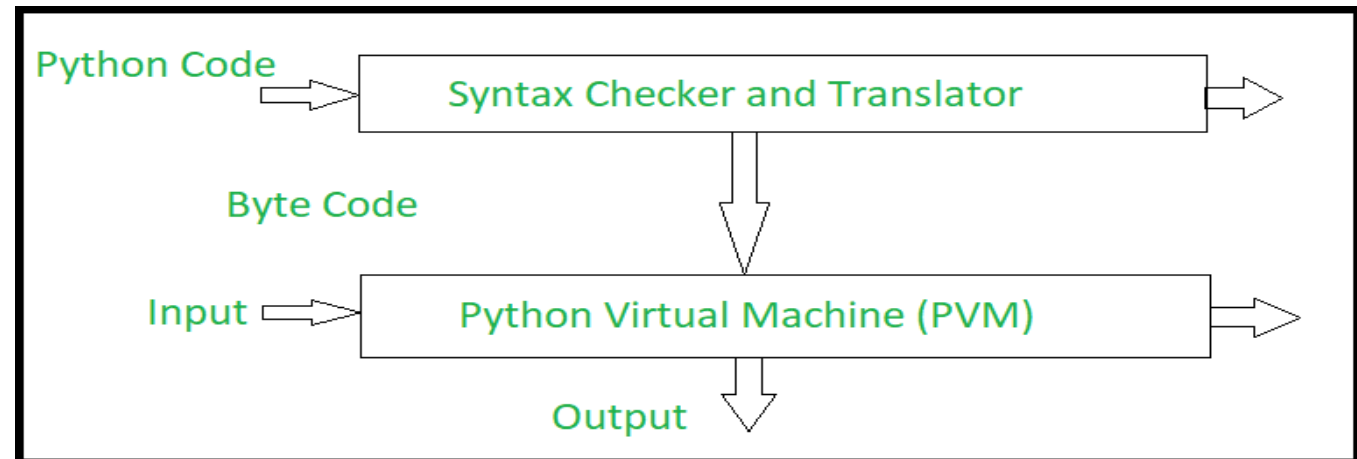
**Disadvantages of interactive mode**

- The interactive mode is not suitable for large programs.
- The interactive mode doesn't save the statements.

# Python Interpreter and its Working

## What is Python Interpreter:-

- It is a one kind of program that reads the code and executes it one by one.
- It converts the code into a language that is compatible with computer hardware.
- Python doesn't convert its code into machine code, something that hardware can understand.
- It actually converts it into a byte code, which creates a file with extension .pyc or .pyo





- The byte code compilation happened internally, and almost completely hidden from developer.
- So within python, compilation happens, but its just not into a machine language. Its into a byte code. This byte code can't be understood by the CPU.
- So we need an interpreter called Python virtual machine to executes the byte code.
- This byte code translation is performed for faster execution, byte code can be run much faster than the original source code statements.

**Note:-**Actually Byte code was developed for computer. Computer easily read this byte code. So, execution is faster.

**The Python source code goes through the following to generate an executable code :**

**Step 1:**

- The python compiler reads a python source code or instruction.
- Then it verifies that the instruction is well-formatted, i.e. it checks the syntax of each line.
- If it encounters an error, it immediately stop the translation and shows an error message.

**Step 2:**

- If there is no error, source code is well-formatted then the compiler translates it into “Byte code”.

### Step 3:

- Byte code is then sent to the Python Virtual Machine(PVM) which is the python interpreter.
- PVM converts the python byte code into machine-executable code.
- If an error occurs during this interpretation, then the conversion is stopped with an error message.

# Syntax and Semantics



- The syntax of a programming language refers to the order to which different elements are combined to form valid expressions.
  - These elements may be words, operators, or phrases.
  - The syntax of the Python programming language is the set of rules that defines how a Python program will be written and interpreted
  - The Python language has many similarities to Perl, C, and Java
- Semantics emphasizes the meaning of a program, so it'll be understandable and easy to predict the outcome of execution.
  - Semantics provides significant information needed to understand a program

# The *foreign language* of python...

syntax

How it looks

semantics

What it does

intent

What it should do



```
name = raw_input('Hi... what is your name? ')
print                                     # prints a blank line

if name == 'Eliot' or name == 'Ran':
    print 'I\'m "offline." Try later.'

elif name == 'Zach':                    # is it Zach?
    print 'Zach Quinto...?', 'No?', 'Oh.'

else:                                   # in all other cases...
    print 'Welcome', name, '!'
    my_choice = random.choice( [ 'R', 'P', 'S' ] )
    print 'My favorite object is', my_choice, "!"
```

```
void foo(int x)
{
    if (x == 0) {
        bar();
        baz();
    } else {
        qux(x);
        foo(x - 1);
    }
}
```

```
1  def foo(x):
2      if x == 0:
3          bar()
4          baz()
5      else:
6          qux(x)
7          foo(x - 1)
~
~
~
~
```

```
name="code leaks"
if name == "code leaks":
    print("hello")
else:
    print("who?")
```

File "<ipython-input-4-0c3ad72d11a0>", line 3

```
    print("hello")
    ^
```

**IndentationError:** expected an indented block

# Variable

- Variables are containers for storing data values.

## Creating Variable:-

- Python has no command for declaring a variable.
- A variable is created at the moment you first assign a value to it.

Example:-

```
x=5
```

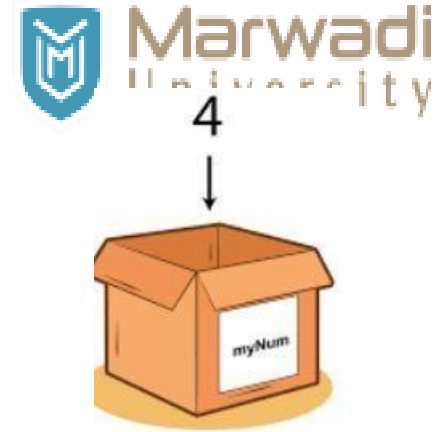
```
print(x)
```

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

```
x=4    #x is of type int
```

```
x="MU Rajkot"  #x is of type str
```

```
print(x)
```



## Get the Type:-

- You can get the type of variable with the type() function.

```
x=5
```

```
y="MU Rajkot"
```

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

## Single or Double Quotes?

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

```
x="MU Rajkot"
```

is same as

```
x='MU Rajkot'
```

## Case-Sensitive

- Variable names are case-sensitive.



x=10

X= "MU Rajkot"

#X will not overwrite x

## **Variable Name**

- A variable can have a short name (like x and y) or a more descriptive name (age, firstName, roll\_no).

## **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.

Legal Variable names:-first\_name, firstName, \_firstName, FIRSTNAME, firstName2

Illegal Variable names:-2firstName, first-name, first name

## Multi Words Variable Names

- Variable names with more than one word can be difficult to read.

```
firstname='Rajesh'
```

```
mycollegename='MU Rajkot'
```

- There are several techniques you can use to make them more readable.

Camel Case:-Each word, except the first, starts with a capital letter.

```
myCollegeName="MU Rajkot"
```

Pascal Case:-Each word starts with a capital letter.

```
MyCollegeName="MU Rajkot"
```

## One Value to Multiple Variables

- You can assign the same value to multiple variables in one line:

```
x=y=z="Engineering"
```

## Many Values to Multiple Variables

- Python allows you to assign values to multiple variables in one line.

```
x, y, z = "Engineering", "Management", "Science"
```

- Make sure the number of variables matches the number of values, or else you will get an error.

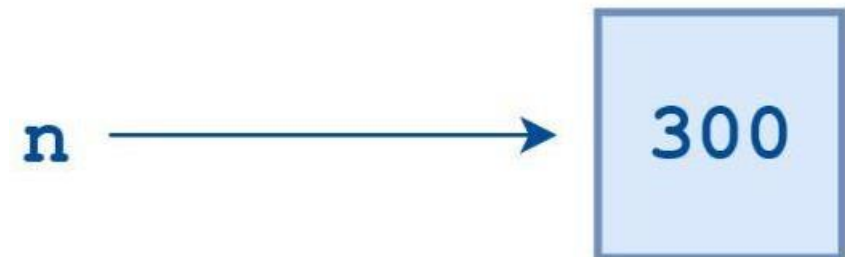
## Object Reference

- A python variable is a symbolic name that points to an object.
- Once an object is assigned to a variable, we can refer to the object by that name.

Example:

```
n=300
```

- This assignment creates an integer object with value 300 and assign the variable n to that object.



- Now Consider the following statement

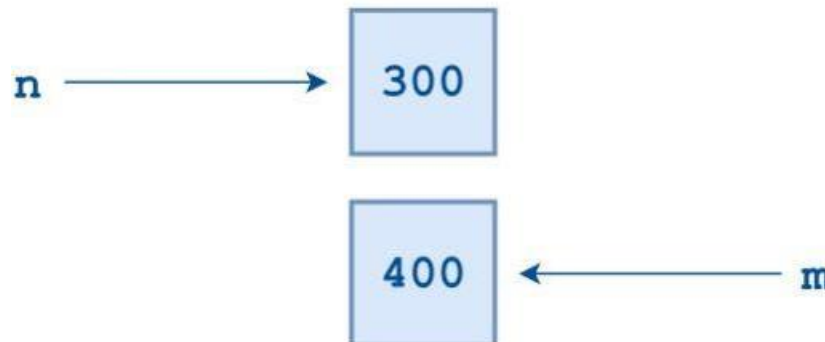
`m=n`



- What happens when it is executed? Python does not create another object.
- It simply creates a new symbolic name or reference, m, which points to the same object that n points to.
- Next, suppose we do this:

`m=400`

- Now Python creates a new integer object with the value 400, and m becomes a reference to it.

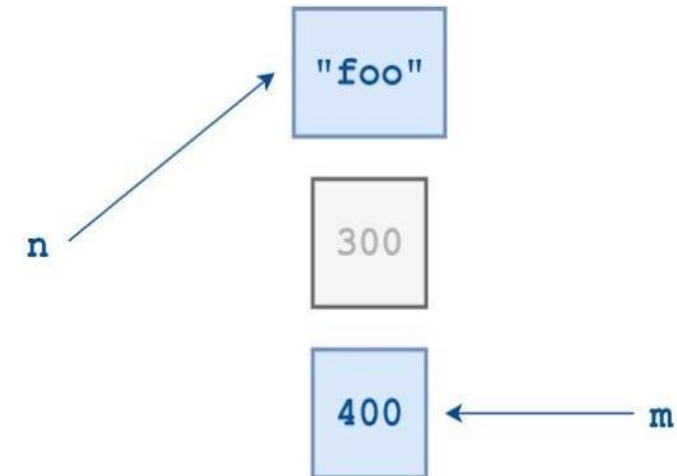


```
n="foo"
```

- Now Python creates a string object with the value "foo" and makes n reference that. There is no longer any reference to the integer object 300.

### Object Identity:-

- In Python, every object that is created is given a number that uniquely identifies it.
- It is guaranteed that no two objects will have the same identifier.
- The built-in Python function `id()` returns an object's integer identifier.



Example:-

```
m=30
```

```
n=30
```

```
print(id(m))
```

```
1405569120
```

```
print(id(n))
```

```
1405569120
```

- Here, m and n are separately assigned to integer objects having value 30.
- But in this case, id(m) and id(n) are same.

# Mutable and Immutable Variable



## Immutable Variable:-

- It is an variable whose value cannot change.
- Therefore, once we assign them the value during declaration we cannot make changes in the future.
- If we try to change immutable variable python gives an error.
- Example of immutable variables in python are int, float, bool, complex, strings, tuples and sets.

Example:-

```
tuple1 = ( 1, 33, 'hello', 45.3 )  
print (tuple1)
```

```
tuple1 [0] = 22  
print (tuple1)
```

TypeError: 'tuple' object does not support item assignment

- Hence, we cannot assign a new value to an old tuple else python gives an error.

### **Mutable Variable:-**

- It is an variable whose value can change.
- Therefore, once we assign them the value during declaration we can make changes in the future.
- Examples of mutable variables in Python are List and Dictionary.



Example:-

```
list1 = [ 1, 33, 'hello', 45.3 ]
```

```
print (list1)
```

```
[ 1, 33, 'hello', 45.3 ]
```

```
list1 [0] = 'hi'
```

```
print (list1)
```

Output:-

```
['hi', 33, 'hello', 45.3]
```

# Data Types

- A variable can hold different types of values.
- Every value in Python has a datatype.
- For example, a person's name must be stored as a string whereas its id must be stored as an integer.
- Data types are actually classes and variables are instance (object) of these classes.
- There are various data types in Python.

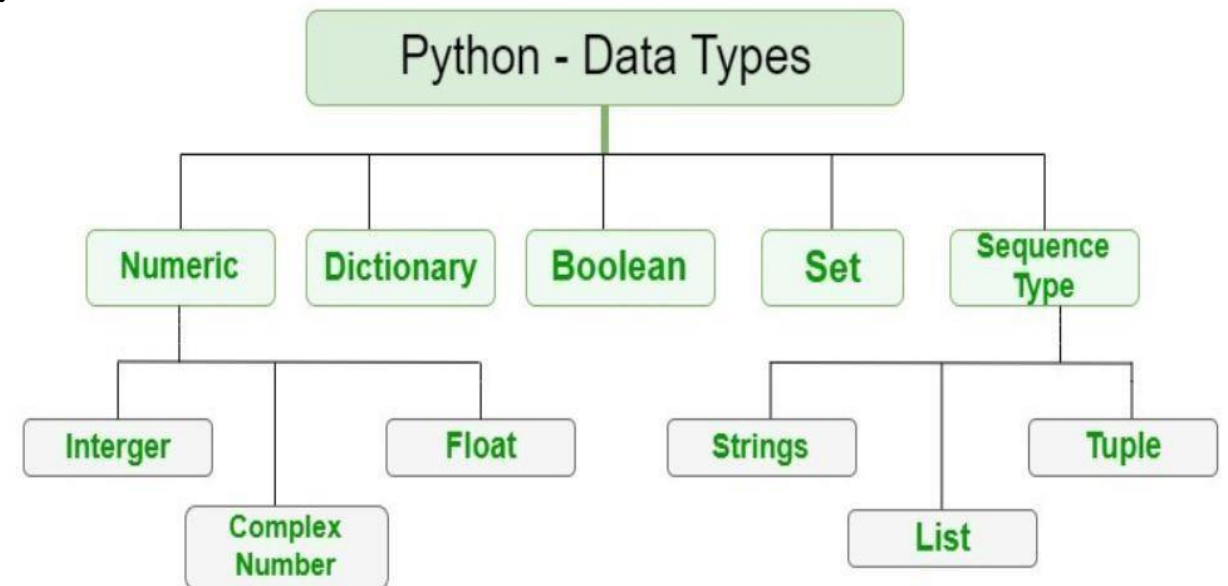
1.Numbers

2.Sequence Type

3.Boolean

4.Set

5.Dictionary



## Numbers

- Number stores numeric values.
- Integers, float and complex numbers belong to a Python Numbers data-type.
- Python provides the `type()` function to know the data-type of the variable.
- Similarly, the `isinstance()` function is used to check if an object belongs to a particular class.

### **int**

- Integer value can be any length such as integers 10, 2, 29, -20, -150 etc.
- Python has no restriction on the length of an integer.

Example:-

```
a=5
```

```
print("The type of a",type(a))
```

Output:-The type of a <class 'int'>

## **float**

- It is used to store floating-point numbers like 1.9, 9.902, 15.2, etc.
- It is accurate upto 15 decimal points.

Example:-

```
b=40.5
```

```
print("The type of b",type(b))
```

Output:-The type of b <class 'float'>

## Complex

- Complex numbers are written in the form,  $x + yj$ , where  $x$  is the real part and  $y$  is the imaginary part.
- The complex numbers like  $2.14j$ ,  $1+2j$ ,  $2.0 + 2.3j$  etc.

Example:-

```
c=1+3j
```

```
print("The type of c",type(c))
```

```
print(" c is a complex number", isinstance(1+3j,complex))
```

Output:-The type of c <class 'complex'>

c is complex number: True

## Sequence Type

### String

- String can be defined as sequence of character.
- In Python, we can use single, double, or triple quotes to define a string. The String in Single and double quote are same.

a="Marwadi University"

b='Marwadi University'

- Multi-line strings can be denoted using triple quotes, `'''` or `"""`.
- Python provides built-in functions and operators to perform operations in the string.

- The operator + is used to concatenate two or more strings as an operation "Marwadi University"+" Rajkot" returns Marwadi University Rajkot. The operator \* is known as a repetition operator as the operation "Python" \*2 returns 'Python Python'.
- Subset of strings can be taken by using slice operator([] and [:]).

## Boolean

- Python Boolean type has only two possible values:

True

False

- Generally, it is used to represent the truth values of the expressions. For example,  $1==1$  is True whereas  $2<1$  is False.

Example:-

```
a=True
```

```
b=False
```

```
type(a)
```

```
type(b)
```

```
<class 'bool'>
```

```
<class 'bool'>
```

- The output `<class 'bool'>` indicates the variable is a boolean data type.
- Note the keywords `True` and `False` must have an Upper Case first letter. Using a lowercase `true` returns an error.
- Integers and floating point numbers can be converted to the boolean data type using Python's `bool()` function.
- An `Int` or `float` number set to zero returns `False`. An `Int` or `float` set to any other number positive or negative returns `True`.



# Comments

- Comments are descriptions that help programmers to understand the code very carefully.
- They are completely ignored by the Python interpreter.
- There are three types of comments in python.

Single Line Comments

Multiline Comments

Docstring Comments

## Single Line Comments:-

- It Starts with hashtag symbol(#).

Example:-

```
#printing college Name  
print("Marwadi Uni.Rajkot")
```

Output

Marwadi Uni.Rajkot

- Here the comment is

#printing a college name

- This line is ignored by the Python interpreter.
- Everything that comes after # is ignored. So, we can write the above program in single line as:

```
print("Marwadi Uni.Rajkot") #printing college Name
```

- If the comment exceeds one line then put a hashtag on the next line and continue the comment.
- It is useful for explaining variables, function declarations and expressions.

## Multi-Line Comment:-

- Python does not provide the option for multiline comments.
- There are three different ways through which we can write multiline comments.

### Using Multiple Hashtags(#):-

- To add a multiline comment we could insert # for each line.

```
#Write a program using a while loop that asks the user for a  
#number, and prints a countdown from that  
#number to zero.
```

### Using String Literals:-

- Python ignores the string literals that are not assigned to a variable so we can use these string literals as a comment.

```
'Write a Python program which checks whether given number is odd or even'
```

- On executing the above code, we can see that there will not be any output.

### Multi-Line comments using String Literals:-

- We can use triple quote to write multiline comments.

```
"""Write a Python Program which checks whether  
given number is odd or even """
```

- Here, the multiline string isn't assigned to any variable, so it is ignored by the interpreter.

## Python docstrings:-

- docStrings are basically multiline comments inside functions, classes, modules.
- docstrings can be accessed using the `__doc__` attribute.

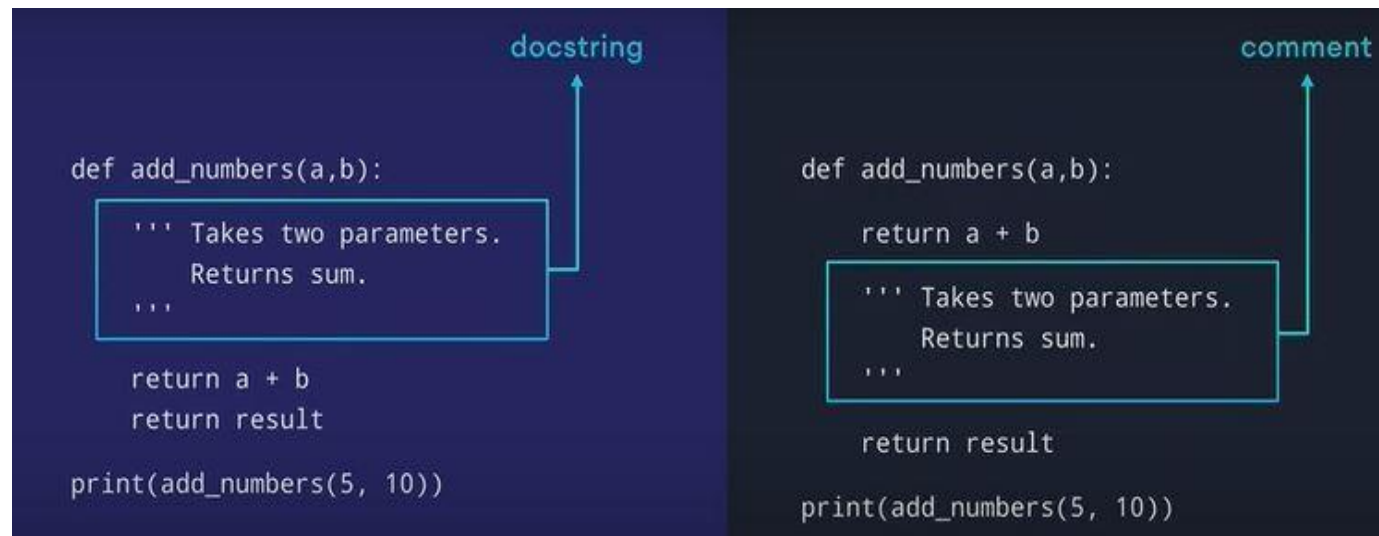
```
def add_numbers(a,b):  
    "Takes two parameters  
    returns its sum"  
    return a+b  
print(add_numbers(4,5))
```

← Python docstring

- `__doc__` is a special attribute that prints a docstrings. If docstrings is not available it prints None.

## How to write good docstrings?

- There are few things we should keep in mind while creating docstrings.
1. Docstrings must be the first statement in a function, otherwise it not considered as docstrings.



2. We can not use regular comments that starts with # to create a docstrings.

```
def add_numbers(a,b):  
    # Takes two parameters. Returns sum.  
    return a + b  
    return result  
  
print(add_numbers(5, 10))
```

not a  
docstring

3. Ending the docstrings with full stop(.)

# Operators

- Operators are used to perform operations on variables and values.
- Python operators can be classified into several categories.

Arithmetic operators

Assignment operators

Comparison operators

Logical operators

Identity operators

Membership operators

Bitwise operators

## **Arithmetic Operators:-**

- Arithmetic operators are used to perform arithmetic operations between two operands.



Operator	Description
+ (Addition)	It is used to add two operands. For example, if $a = 20$ , $b = 10 \Rightarrow a + b = 30$
- (Subtraction)	It is used to subtract the second operand from the first operand. If the first operand is less than the second operand, the value results negative. For example, if $a = 20$ , $b = 10 \Rightarrow a - b = 10$
/ (divide)	It returns the quotient after dividing the first operand by the second operand. For example, if $a = 20$ , $b = 10 \Rightarrow a / b = 2.0$
* (Multiplication)	It is used to multiply one operand with the other. For example, if $a = 20$ , $b = 10 \Rightarrow a * b = 200$
% (reminder)	It returns the reminder after dividing the first operand by the second operand. For example, if $a = 20$ , $b = 10 \Rightarrow a \% b = 0$
** (Exponent)	It is an exponent operator represented as it calculates the first operand power to the second operand.
// (Floor division)	The division of operands where the result is the quotient in which the digits after the decimal point are removed.

## Comparison Operators:-

- Comparison operators are used to compare two values:

Operators	Description
==	If the value of two operands is equal, then the condition becomes true.
!=	If the value of two operands is not equal, then the condition becomes true.
>	If the value of left operand is greater than the value of right operand, then condition becomes true.
<	If the value of left operand is less than the value of right operand, then condition becomes true.
>=	If the value of left operand is greater than or equal to the value of right operand, then condition becomes true.
<=	If the value of left operand is less than or equal to the value of right operand, then condition becomes true.

## Logical Operators:-

- Logical operators are used to combine conditional statements.

Operator	Description
and	If both the expression are true, then the condition will be true.
or	If one of the expressions is true, then the condition will be true.
not	If an expression is true, then result will be false and vice versa.

## Identity Operators:-

- Identity operators compares the memory locations of two objects.
- There are two Identity operators like is and is not.

Operator	Description
is	True if the operands are similar
is not	True if the operands are not similar

### **Membership operator:-**

- in and not in are the membership operators; used to test whether a value or variable is in a sequence.

Operator	Description
in	True if value is found in the sequence
not in	True if value is not found in the sequence

## Bitwise Operators:-

- It is used to compare binary numbers.

Operator	Description
&	Sets each bit to 1 if both bits are 1
	Sets each bit to 1 if one of two bits is 1
^	Sets each bit to 1 if only one of two bits is 1
~	Inverts all the bits
<<	The left operand value is moved left by the number of bits present in the right operand.
>>	The left operand is moved right by the number of bits present in the right operand.

## Assignment Operators:-

Operators	Description
=	Assigns values from right side operands to left side operand
+=	It adds right operand to the left operand and assign the result to left operand
-=	It subtracts right operand from the left operand and assign the result to left operand
*=	It multiplies right operand with the left operand and assign the result to left operand
%=	It takes modulus using two operands and assign the result to left operand
**=	Performs exponential (power) calculation on operators and assign value to the left operand
//=	It performs floor division on operators and assign value to the left operand

# String Operations

## **find():-**

- It finds substring in the whole string and returns index of the first match.
- It returns -1 if substring does not match.

**Syntax:-**`string.find(value,start,end)`

**value:-**It is the substring that needs to be searched in the given string.

**start:-**Where to start the search. Default is 0

**end:-**Where to end the search. Default is to the end of the string

**Example:-**

```
txt="Javascript is a scripting is language"
```

```
print(txt.find("is"))
```

```
print(txt.find("IS"))
```

```
print(txt.find("is",15))
```

```
print(txt.find("z"))
```

## **index():-**

- The index() method returns the index of a substring inside the string (if found). If the substring is not found, it raises an exception.

**Syntax:-**`string.index(value,start,end)`

**value:-**It is the substring that needs to be searched in the given string.

**start:-**Where to start the search. Default is 0

**end:-**Where to end the search. Default is to the end of the string

**Note:-**The find() method is almost the same as the index() method, the only difference is that the index() method generates an error if the value is not found and find() method returns -1 if the value is not found.



## **format():-**

- The format() method formats the specified value(s) and insert them inside the string's placeholder.
- The placeholder is defined using curly brackets: { }.
- The format() method returns the formatted string.

**Syntax:-**string.format(value1, value2...)

value1,value2:-Required. One or more values that should be formatted and inserted in the string. The values are either a list of values separated by commas, a key=value list, or a combination of both. The values can be of any data type.

- The placeholders can be identified using named indexes {firstName}, numbered indexes {0}, or even empty placeholders { }.

Example:-

```
txt1="My name is {fname}, I'm {age}".format(fname="Vaibhav",age=30)
```

```
txt2="My name is {0}, I'm {1}".format("Vaibhav",30)
```

```
txt3="My name is {}, I'm {}".format("Vaibhav",30)
```

Output:-

My name is Vaibhav I'm 30

My name is Vaibhav I'm 30

My name is Vaibhav I'm 30

### **isalpha():-**

- The isalpha() method returns True if all the characters are alphabet letters (a-z). If not, it returns False.

syntax:-string.isalpha()

- It doesn't take any parameters.

### **isdigit():-**

- The isdigit() method returns True if all the characters are digits, otherwise False.

syntax:-string.isdigit()

- The isdigit() doesn't take any parameters.

### **isalnum():-**

- The isalnum() method returns True if all characters in the string are alphanumeric (either alphabets or numbers). If not, it returns False.

**syntax:-**`string.isalnum()`

- It doesn't take any parameters.

**startswith():-**

- The startswith() method returns True if the string starts with the specified value, otherwise, False.

**syntax:-**`string.startswith(value, start, end)`

**value:-** The value to check if the string starts with

**start:-** Beginning position where value(string) is to be checked within the string.

**end:-** Ending position where value(string) is to be checked within the string.

**Example:-**

```
text = "Python programming is easy."  
text.startswith('programming is')
```

```
text.startswith('programming is',7)
```

```
text.startswith('programming is',7,18)
```

```
text.startswith('program', 7, 18)
```

### **endswith():-**

- It returns True if the string ends with the specified value, otherwise False.

**syntax:-**`string.endswith(value, start, end)`

**value:-**The value to check if the string ends with

**start:-**Beginning position where value is to be checked within the string.

**end:-**Ending position where value is to be checked within the string.

Example:-

```
text = "Python programming is easy to learn."
```

```
text.endswith('is', 7, 26)
```

```
text.endswith('easy', 7, 26)
```

**isupper():-**

- It returns True if all the characters are in upper case, otherwise False.
- Numbers, symbols and spaces are not checked.
- The isupper() method doesn't take any parameters.

Example:-

```
str="MARWADI UNIVERSITY"
```

```
print(str.isupper())
```

Output:-

True

Example:-How to use isupper() in a program

```
str="MARWADI UNIVERSITY"
```

```
if str.isupper()==True:
```

```
    print("Does not contain lowercase letter")
```

```
else:
```

```
    print("Contains lowercase letter")
```

**zfill():-**

- The zfill() method returns a copy of the string with '0' characters padded to the left.
- The width specifies the length of the returned string with 0 digits filled to the left.

**Syntax:** `string.zfill(width)`

- If the value of the width parameter is less than the length of the string, no filling is done.

**Example:-**

```
text="python programming"  
print(text.zfill(20))  
print(text.zfill(15))  
print(text.zfill(len(text)+3))
```

**Output:-**

```
00python programming  
python programming  
000python programming
```



- If a string starts with the sign ('+', '-'), 0 digits are filled after the first sign prefix character.

### **join():-**

- It returns a string by joining all the elements of list or tuple separated by a string separator.

**syntax:-separator.join(list/tuple)**

- If the list or tuple contains any non-string value, it generates a TypeError exception.

```
txt=["10","vaibhav","harsh","89"]
```

```
print(' '.join(txt))
```

```
print(' , '.join(txt))
```

## **replace():-**

- The replace() method replaces a group of words with another group of words.
- All occurrences of the specified words will be replaced, if nothing else is specified.

**syntax:-**`string.replace(oldvalue, newvalue, count)`

**oldvalue:-**The string to search for

**newvalue:-**The string to replace

**count:-**A number specifying how many occurrences of the old value you want to replace. Default is all occurrences

```
txt="Javascript is a scripting is language is"
```

```
print(txt.replace("is","are"))
```

```
print(txt.replace("is","are",2))
```

## **split():-**

- The split() method breaks up a string at the specified separator and returns a list.

**syntax:-**`string.split(seperator,maxsplit)`

**seperator:-**Delimiter at which splits occur. If not provided, the string is splitted at whitespaces.

**maxsplit:-**Specifies how many splits to do. Default value is -1, which is "all occurrences"

- If maxsplit is specified, the list will have a maximum of maxsplit+1 items.

`txt="javascript is scripting language"`

`txt.split(' ')`

`txt.split(' ',2)`

## **splitlines():-**

- It splits the string at line breaks and returns a list.

Example:

```
a="BTech CE\nMarwadi University\nRajkot"
```

```
print(a.splitlines())
```

Output:

```
['BTech CE', 'Marwadi University', 'Rajkot']
```

**Syntax:-**`string.splitlines(keeplinebreaks)`

**keeplinebreaks:-**Optional. Specifies if the line breaks should be included(True) or not(False).Default is False.

- If there are no line break characters, it returns a list with a single item (a single line).

- It takes an integer value as parameter. Here 0 represents False and other positive or negative numbers indicate True.

### **lower():-**

- The lower() method returns a string where all characters are lower case.
- Symbols and Numbers are ignored.

### **Syntax:-string.lower()**

- It doesn't take any parameters.
- If no uppercase characters exist, it returns the original string.

### **upper():-**

- The upper() method returns a string where all characters are in upper case.
- Symbols and Numbers are ignored.

### **Syntax:-string.upper()**

- It doesn't take any parameters.
- If no lowercase characters exist, it returns the original string.

### **title():-**

- The title() method returns a string where the first character in every word is upper case.
- If the word contains a number or a symbol, the first letter after that will be converted to upper case.
- It doesn't take any parameters.

### **capitalize():-**

- It converts the first character of a string to an uppercase letter and all other alphabets to lowercase.

**Syntax:-**string.capitalize()

- This method doesn't take any parameter.
- It returns a new string and doesn't modify the original string.

### **isprintable():-**

- It returns True if all the characters in string are printable. If not, it return False.

Note:-

Printable Characters:-symbols,digits,comma,parentheses,point,whitespace

Non-printable Characters:-Line breaks

Example:

```
str="Rajkot"
```

```
str1="Rajkot\n"
```

```
print(str.isprintable())
```

```
print(str1.isprintable())
```

Output:

True

False

- The `isprintable()` method doesn't take any parameters.
- The `isprintable()` method returns True when we pass an empty string.

**`isnumeric()`**:-

- The `isnumeric()` method returns True if all the characters are numeric (0-9), otherwise False.
- The `isnumeric()` method doesn't take any parameters.

Example:-

```
str="3467"
```

```
str1="3467R"
```

```
print(str.isnumeric())
```

```
print(str1.isnumeric())
```



## **isspace():-**

- It returns “True” if all characters in the string are whitespace characters, Otherwise, It returns “False”.
- Characters that are used for spacing are called whitespace characters. For example: tabs(\t), spaces(‘ ‘), newline(\n), etc.

Whitespace character:-

\n:-New Line

\t:-Horizontal Tab

‘ ‘:-Space

**Syntax:-**string.isspace()

Example:-

```
str="  \t "
```

```
print(str.isspace())
```

```
str=" R "
```

```
print(str.isspace())
```

**istitle():-**

- The istitle() method returns True if all words in a text start with a upper case letter, AND the rest of the word are lower case letters, otherwise False.
- Symbols and numbers are ignored.
- The istitle() method doesn't take any parameters.

**Example:-**

```
str="Marwadi University Rajkot @3"  
print(str.istitle())
```

Output:-

True

**Example:-**How to use istitle()?

```
str="Marwadi University Rajkot"  
if str.istitle() == True:  
    print('Titlecased String')  
else:
```

```
    print('Not a Titlecased String')
```

Output:-

Titlecased String

## **lstrip():-**

- It is used to remove all leading characters from string.

### **Syntax:-string.lstrip(chars)**

- It takes char type parameter which is optional. If parameter is not provided, it removes all leading spaces from the string.
- It returns a copy of the string with leading characters stripped.
- All combinations of characters in the chars argument are removed from the left of the string until the first mismatch.

### **Example:**

```
str="  Marwadi University  "
```

```
print(str)
```

```
print(str.lstrip())
```

Output:

Marwadi University

Marwadi University

Example:

```
str="$$$$-Rajkot-$$$$"
```

```
str1="https://www.gtu.ac.in"
```

```
print(str.lstrip('$'))
```

```
print(str1.lstrip('https://w.'))
```

Example:

```
str=' this is good '
```

```
print(str.lstrip())
```

```
print(str.lstrip('s ti'))
```

```
print(str.lstrip('sti'))
```

## **rstrip():-**

- It removes all the specified characters from right side of string.
- If we don't specify the parameter, It removes all the whitespaces from the string.
- It returns a string value.

**Syntax:** `string.rstrip(chars)`

**Example:-**

```
str=" this is good "
```

```
print(str.rstrip())
```

```
print(str.rstrip('sioo'))
```

```
print(str.rstrip('sid oo'))
```

Output:-

this is good

this is good

this is g

**strip():-**

- It removes characters from both left and right end of the string by specifying a set of characters to strip() function as an argument.
- By default, it removes whitespaces from starting and ending string if no argument is passed to the strip() function.

Syntax:-string.strip(chars)

Example:-

```
str="  xoxo Rajkot xoxo  "  
print(str)#len(str)  
print(str.strip()) #len(str.strip())  
print(str.strip(' xot'))  
print(str.strip('xot'))
```

Output:-

```
  xoxo Rajkot xoxo  
xoxo Rajkot xoxo  
Rajk  
  xoxo Rajkot xoxo
```



Thank You

