**PYTHON NOTES**

# Difference between Compiler and Interpreter.

* Compiler compiles the code all at once and throws errors while interpreter compiles the code line by line. Interpreter follows synchronous compilation.

# Code execution in Python:

* The code written which is High level language is converted into a machine-readable binary code. This binary code is now passed to a virtual machine which is known as Python Virtual Machine for Python that checks if there are any errors in the code and reports them before passing it to the machine which executes the code directly without compiling it as the code will be free of errors because of Python Virtual machine.

# Difference between CLI,IDE and IDLE:

* CLI is nothing but Command Line Interface where the execution happens for every line and the user cannot save or reuse the file.
* IDE is Integrated Development Environment and IDLE is Integrated Development Learning Environment. IDE will have auto suggestion feature and a lot of other libraries that makes it easier for the developers to work on the code. You can have multiple tabs and work on many work items simultaneously.
* In IDLE, you can use it for learning purposes where you run a piece of code and check if it is generating the desired output. It doesn’t have all the features that an IDE has.
* **Example**: PyCharm is an IDE while Python IDLE is an example of IDLE.

# Integrated IDEs/ Virtual Environments:

* A collection of multiple IDEs in one place is called Integrated IDEs.
* Example: Anaconda Prompt is an example of Integrated IDEs where you have Jupyter Notebook, VS code in a single environment.
* Virtual Environments is built somewhere, and we will use it based on our needs. On some cases, when you use higher libraries in python, All the functionalities will not be available under the same IDE.

# Evolution of IDE:

* CLI🡪IDLE🡪IDE🡪Integrated IDEs(Virtual Environment).

# Synchronous Compilation and Asynchronous compilation:

* Synchronous compilation will not execute the second line of code before doing the first while Asynchronous compilation does all the lines of code together.

# String:

* String is enclosed within single or double quotes.
* There are lot of built in functions or multiple ways to accessing the string.

## Slicing:

* + Slicing is a way to access through the strings.
  + Every character starts with an index ‘0’ and goes till length of the string – 1.
  + Negative indexing – The last character is indexed as ‘-1’ and it goes as -2,-3 towards the right.
  + Syntax:
    - String[start position: end position: increment/decrement]
      * Start position – start of the string(i.e from the left it is always starts with ‘0’ and from the right it is always ‘-1’.
      * End position – the index until which you want to traverse through the string. It is usually till any specific index or till the length of the string.
      * Increment/Decrement – The step count. If it is negative, then the string is traversed from the last.
* Use of Slice function: String[slice(start position: end position: increment/decrement)]
  + Start position – start of the string(i.e from the left it is always starts with ‘0’ and from the right it is always ‘-1’.
  + End position – the index until which you want to traverse through the string. It is usually till any specific index or till the length of the string.
  + Increment/Decrement – The step count. If it is negative, then the string is traversed from the last.

Notes:

* You cannot print the last character on giving negative index for both start and end pos.

Examples of Slicing:

A screenshot of a computer program

Description automatically generated

## Built in Functions:

### StartsWith():

There are three types of using startswith() function which is as follows with syntax.

Syntax:

Method 1 - string.startswith(substring(required))

Method 2 - string.startswith(substrig,start\_pos(optional),end\_pos(optional))

Method 3 - string.startswith(tuple(strings),start\_pos(optional),end\_pos(optional))

Note: we can also give a tuple of strings and check if the string startswith any one substring of that given tuple

Example:

A screen shot of a computer program

Description automatically generated

### EndsWith():

There are three types of implementing endsWith().

Syntax:

Method 1 - string.endswith(substring(required))

Method 2 - string.endswith(substring,start\_pos(optional),end\_pos(optional))

Method 3 - string.endswith(tuple(strings),start\_pos(optional),end\_pos(optional))

we can also give an tuple of strings and check if the string endswith any one substring of that given tuple

**Example:**

A computer screen shot of text

Description automatically generated

### Upper():

Converts the string to upper case

### Lower():

Converts the string to lower case.

### Capitalize():

Converts the first character of the string to upper case.

### Title():

Converts the first character of each word in the string.

### Swapcase():

Converts the upper to lower case and lower case to upper case.

Example:

A screenshot of a computer program

Description automatically generated

### Strip():

This method is used to remove white spaces in the string.

#### LStrip():

This method is used to remove the white spaces in the string from the left-hand side.

#### RStrip():

This method is used to remove the white spaces in the string from the right-hand side.

Example:

A screenshot of a computer program

Description automatically generated

Output:

A screen shot of a computer

Description automatically generated

### Split():

Syntax:

* string.split(sub\_string)
* string.split(sub\_string,max\_split(optional))

Additional info:

* split and rsplit functionalites and syntax is almost same.
* split with max\_split will only differ in rsplit , rest all other rsplit logics are same as split()
* split with max\_split will start from leftmost in split whereas it will start from right by default in rsplit().

Notes:

* if we give split without any split condition it will split default with whitespace.
* if there is no split condition present in string , it will return only the single string will be
* present in the list.

Example:

Split and RSplit():

#### RSplit():

A screenshot of a computer program

Description automatically generated

Output:

A screen shot of a computer code

Description automatically generated

#### Splitlines():

Example:

A computer screen shot of text

Description automatically generated

Output:

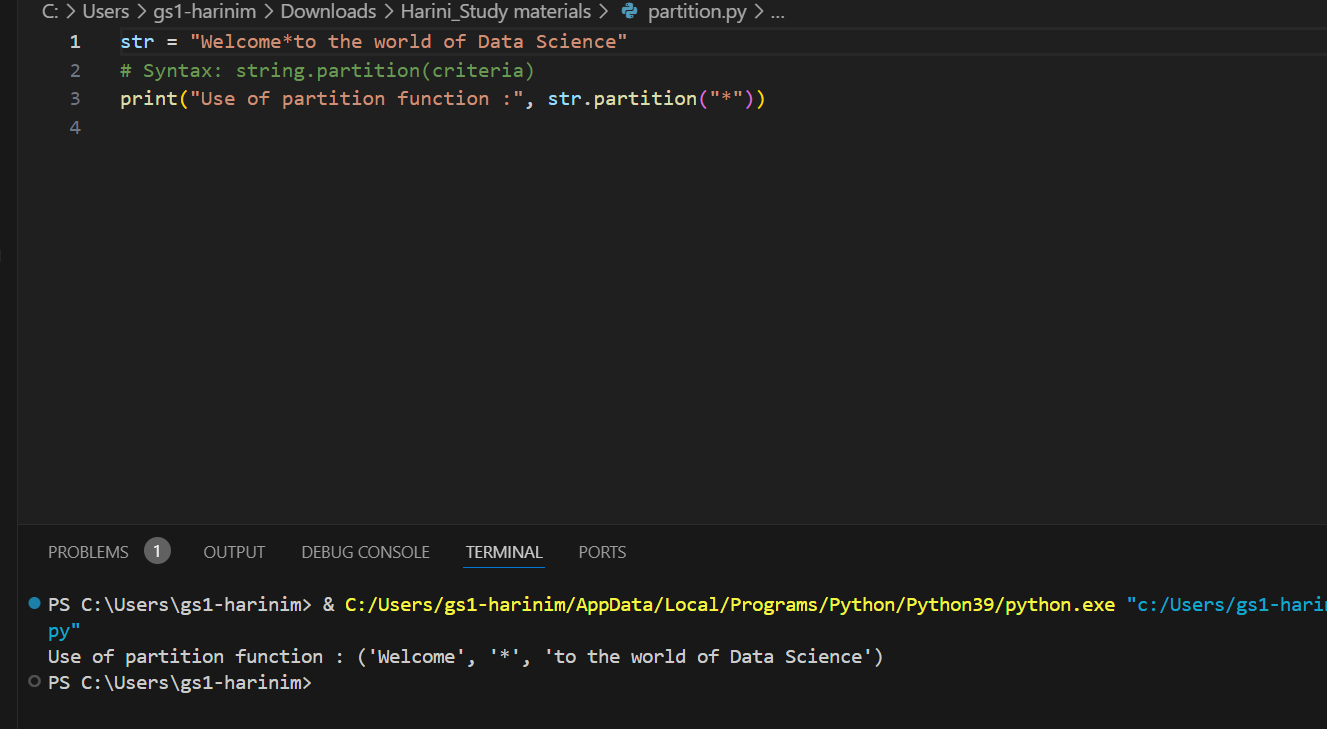
A black screen with white text

Description automatically generated

### Partition():

Partition() is used to split two strings but returns the result as tuple. This splits the first found parameter and returns the left-hand side as one element, the string/param as second element and the right-hand side of param as third element.

Example:



### Find():

* It is used to return the 'index' position of the given substring to be searched.
* If the substring is not present in the string , "-1" will be thrown instead of an error.(but in case of index,rindex error will be thrown which you will see it later)

Example:

Syntax : method 1 - string.find(substring(required))

method 2 - string.find(substrig,start\_pos(optional),end\_pos(optional))

#### Rfind():

Rfind() returns the right most first occurrence of the character in a string and the syntax of rfind() is same as that of find().

Examples:

A screenshot of a computer program

Description automatically generated

Output:

A black screen with white text

Description automatically generated

### Index():

* Index() is used to return the index of the substring in the string. This also returns the first occurrence of the substring from left to right.
* Unlike find(), Index() will throw an error incase if the substring is not present inside the string. This is because Index function is based on ‘AND’ condition while find() is based on ‘IF’ condition.

Syntax:

Method 1 - string.index(substring(required))

Method 2 - string.index(substrig,start\_pos(optional),end\_pos(optional))

#### RIndex():

RIndex() is used to return the index of the substring in the string from the right most side and returns the occurrence of the string from right to left. This also returns the first occurrence of the substring.

Example:

A computer screen shot of text

Description automatically generated

Output:

A computer screen shot of a black screen

Description automatically generated

### Replace:

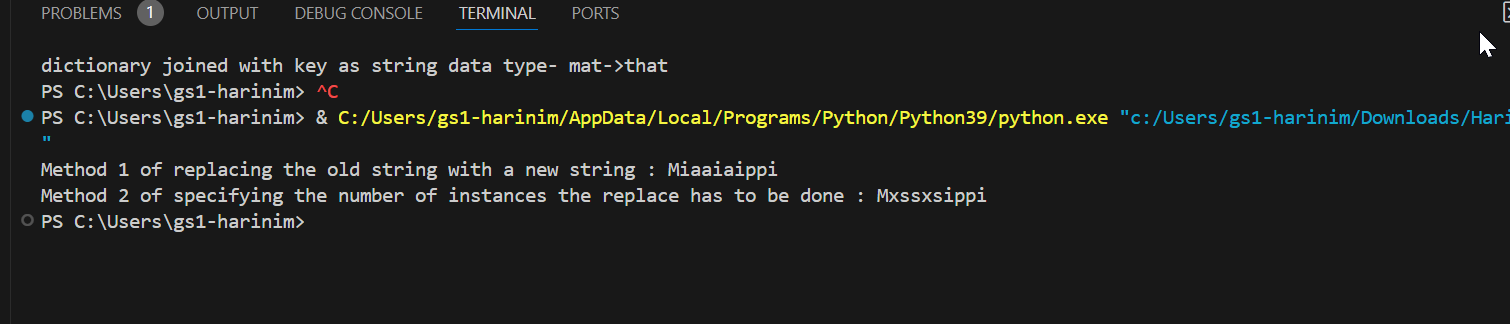
Replace() is used to replace the old string with new string.

Example:

A computer screen with text on it

Description automatically generated

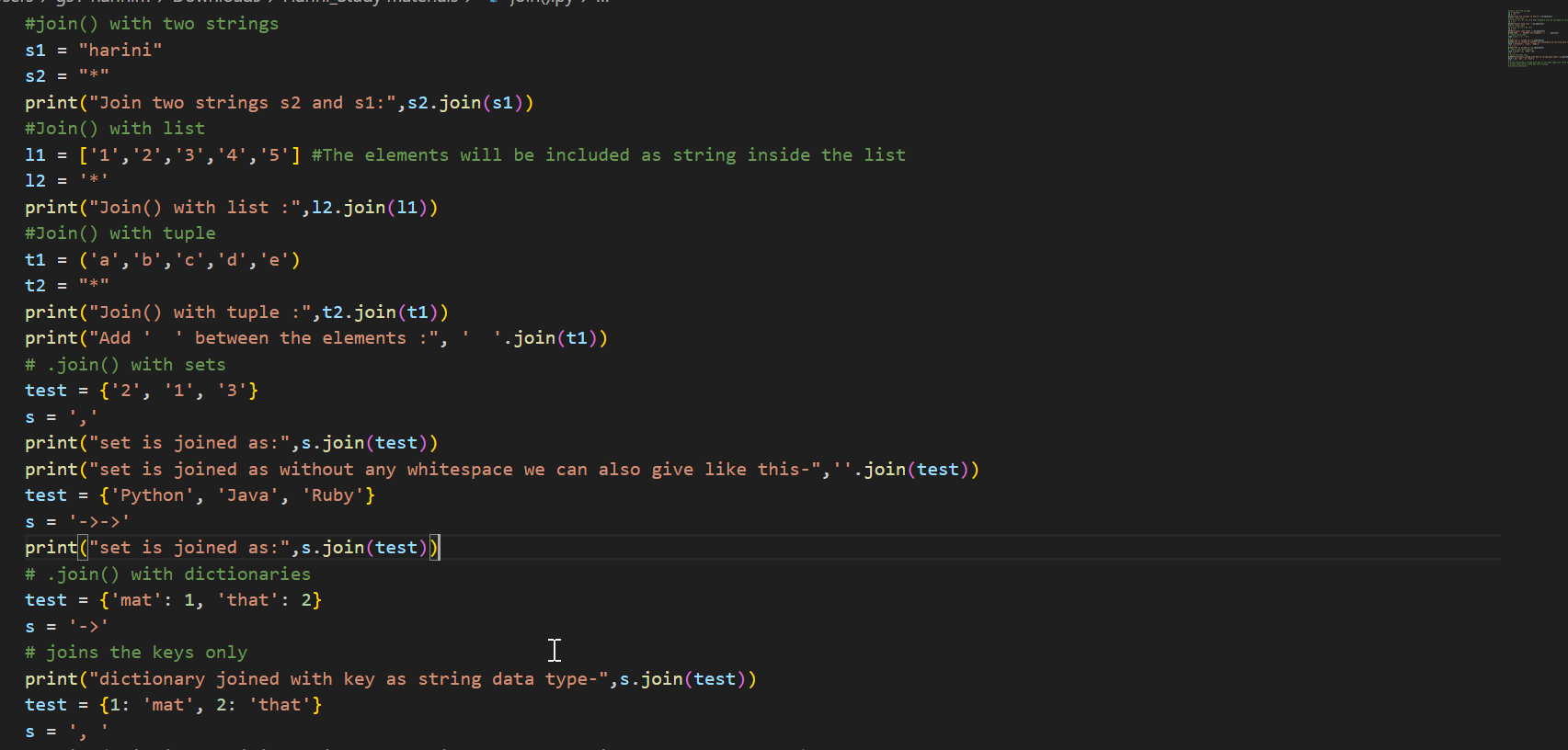
Output:



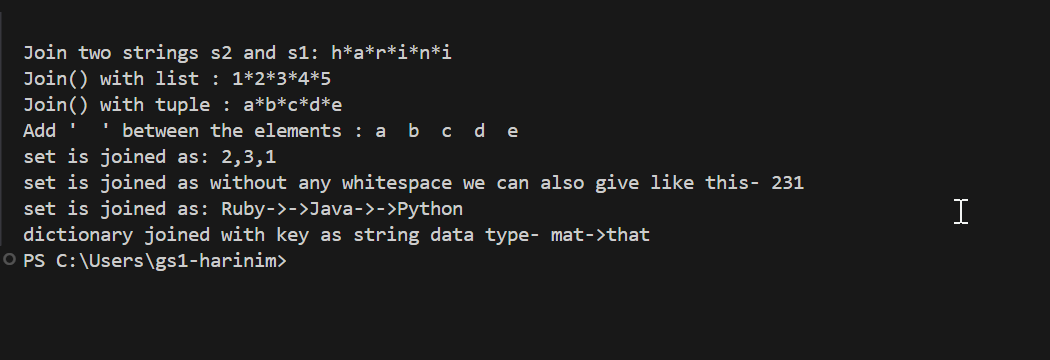
### Join():

Join() is used to combine any iterable for with another string and return type will always be string.

Example:



Output:



### Is():

Any Is() returns Boolean value as the result.

#### IsUpper():

Returns true if the string(all the characters) is in UPPER case and returns false if the string is in LOWER case.

#### IsLower():

Returns true if the string(all the characters) is in LOWER case and returns false if the string is in UPPER case.

#### IsAlpha():

Returns true if all the characters in the string are “ALPHABET” and false otherwise.

#### IsDigit():

Returns true if all the characters in the string are digits.

#### IsDecimal():

Returns true if all the characters in the string are either decimal or number.

#### IsAlnum():

Returns true if all the characters in the string are either Alphabet or Number or contains both.

Example:



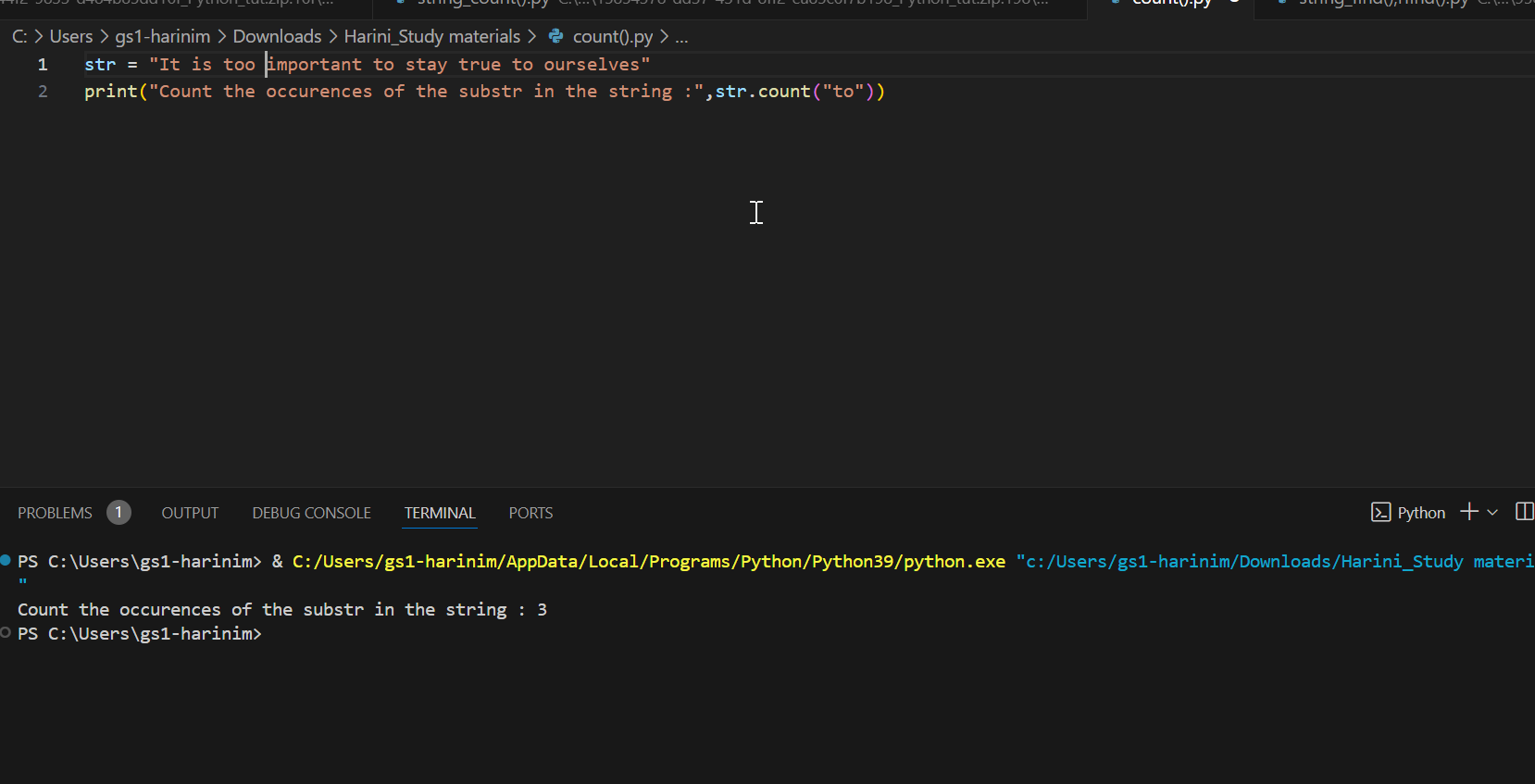
Output:



### Count():

Count() is used to count the occurrences of the substring in the string.

Example:



# Loop:

Loop refers to iterating over an iterable collection of data again and again.

## For loop:

* A for loop is used for iterating over a sequence (that is either a list, a tuple, a dictionary, a set, or a string).
* This is less like the for keyword in other programming languages and works more like an iterator method as found in other object-orientated programming languages.
* With the for loop, we can execute a set of statements, once for each item in a list, tuple, set etc.

Example:

C:\Users\gs1-harinim\Downloads\Harini\_Study materials\for.py

## While loop:

* While loop we can execute a set of statements if a condition is true.
* Unlike for loop, you need to explicitly mention the incremental step here.

Example:

C:\Users\gs1-harinim\Downloads\Harini\_Study materials\while.py

# Break:

It will break the code when the condition is not getting satisfied.

# Continue:

The code will continue to execute irrespective of the condition

# Pass:

It is like continue. The code will pass once the condition is executed. If there is any pending logics to be implemented, then “pass” is used by developers

# Notes:

Why is the initialization and incrementation done explicitly in C and not in Python for loop?

1. In C we follow character by character compilation. On the other hand, Python doesn’t support semi colon and anything unlike and we cannot categorize between condition, initialization and incrementation.

E.g.: In C 🡪for(i=0;i<=n;i++)

If we write the same thing in python,

It will look like 🡪for(i:0 i<n :i++) which is incorrect

1. In C, we compile the code character by character which in simple terms is before the semi colon while in python it is a line-by-line compilation.

## Difference between for and while.

* For loop will have a termination block and While loop won’t have a termination block.
* User defines the condition for while loop and For loop la system will have control over the condition.
* For loop cannot have infinite loop. While can have infinite loop.

## While:

* Unlike for loop, we must initialize and increment the loops.
* Your outputs will be printed only if you initialize the value for i

## Why don’t we have iterator concept in while loop unlike for?

1. In while loop, you will give the control of initialization, incrementation and condition to the user instead of system. In for loop, It is vice-versa

# Lambda:

Lambda is a shorthand of function.

Syntax : lambda arguments : expression.

Arguments – values passed to a function

Expressions – a piece of code that has operations or action of what a function does.

* Lambda is an anonymous function as we do not define any names to Lambda the way we do for functions.
* It is a stateless function. It adapts itself based on the execution. If you give a condition, it acts and executes the function accordingly.

Examples: def add\_num(a, b):

return a + b

In the above example, we can call the add\_num() only if we want to perform addition of two numbers. If you want to multiply two numbers, you should redefine the existing function with another piece of expression. While in Lambda, you can use the same function for performing both addition and multiplication.

* The biggest advantage of lambda is that we do not have separate memory allocation as function has.

Example: def add\_num(a, b):

return a + b

In the above example, when you use function, the memory allocation for the arguments and expressions are done when the function is defined. There is no separate memory allocated whenever the function is called again. In case of Lambda, the memory allocation is done at the time when the lambda function is written, and memory allocation is killed once it is executed.

* The biggest disadvantage of using lambda is that you cannot call lambda function again and again in a job like how you call def function.

# Lists:

## Properties of list:

* List will be surrounded by square brackets.

In other words, List is also an array but not exactly array as list is a step ahead of array. This is because we can store values of different data types in list while array can have values of same type.

* List is mutable which means you can keep changing your list.
* Lists are always ordered.
* Lists allows duplicates.
* List allows data of different data types.

## Accessing lists:

You can access lists using these two methods:

1. Using for loops
2. Slice operator.

A computer code with red and green text

Description automatically generated

Why is list always mutable and Tuple is always immutable?

1. There are so many inbuilt methods in list with which you can modify the list which means you can delete, add, update values to the list.
2. You cannot perform these operations in tuple.

## Nested list:

1. Nested list will be used for cases where the format is in JSON format.
2. Nested list will also be used in cases that are in the form of matrix.

### Disadvantages of a nested list:

When we used nested list, it is difficult to access the list when compared to dictionary.

## Unpacking lists:

Unpacking lists is nothing, but you are going to dissolve the list into separate variables.

### Static way:

L = [1,2,3,4]

a,b,c,d = l

a=1 ,b =2, c=3, d=4

this will return all the individual element of the ist assigned to each variable.

Note: The length of the list and the number of variables on the left hand side should match.

### Dynamic way:

a,b,\*c = l

This will return the first two variables as separate elements, and the c will be populated as a list with all the left values to it

Output: a=1,b=2,c=[3,4]

## In built functions of List:

### Len:

This is used to find the length of a list.

Syntax: len(list)

### Addition to a list:

#### Append :

This function is used to add value to the list, and it always adds the value to the last position of the list. This is used to add element in the list.

**Syntax – list.append(element)**

#### Insert :

This function is used to add value to the list at a specific index.

**Syntax – list.insert(index,element)**

### Deletion of a list:

#### Remove :

This function is used to remove the first occurring element from the list when traversing from left to right.

**Syntax – list.remove(element)**

#### Pop :

This function is used to remove the last element from the list.

**Syntax – list. Pop(index)**

#### Clear :

This function is used to empty the list.

**Syntax – list.clear()**

#### Delete :

This function is used to delete the list itself.

**Syntax – list.delete()**

### Addition of lists itself:

#### Extend :

This function is used to add/concat a list to the existing list.

**Syntax – list1.extend(list2)**

### Aggregate functions:

#### Count :

This is used to count the occurrence of element in a list.

**Syntax – list.count(element)**

#### Index :

This is used to find the index of the element in the list.

**Syntax – list.index(element)**

### Reversing a list:

#### Reverse :

This is used to reverse the original list.

#### Reversed :

This is used to reverse the copy of the original list while the original list remains the same.

### Sorting a list:

#### Sort :

This is used to sort the original list.

#### Sorted:

This is used to sort the copy of the list while the original list remains the same.

Note: You can also use reversed and sort function together when you wanted to find the minimum or maximum in a list.

Eg: list = [1,2,3,4,5,6]

Sorted(list,reversed=”true”)

### Min/Max of the list

#### Min :

to find the minimum of the list.

#### Max :

to find the maximum of the list.

#### Sum :

to sum all the elements of the list.

##### Sum of the string:

A computer screen shot of a code

Description automatically generated

**Example:** C:\Users\gs1-harinim\Downloads\Harini\_Study materials\list.py

# Tuple:

## Properties of Tuple:

1. Tuples are always enclosed within parentheses
2. Tuples are immutable which means we cannot keep changing the list.
3. Tuples are ordered.
4. Tuples allow duplicates.
5. Tuples allow data of different types.

## Accessing the tuple:

1. Tuple can be accessed using index.
2. By using slice operator.
3. By using for loop.

***Note:*** *You can update a tuple by converting into a list first, then perform required CRUD operations to a tuple.*

## Updating Tuples:

1. Converting it to a list.
2. Adding tuple to a tuple.

## Unpacking tuples:

*Same as that of list(Static and dynamic)*

Remember always whenever you unpack a list or tuple, it always returns the list data type.

## Joining Tuples:

1. + operator – joining two tuples (alter of extend function)
2. \* Operator – repeat the tuples.

## Built in functions of Tuples:

### Reversing a tuple:

#### Reverse :

This is used to reverse the original tuple.

#### Reversed :

This is used to reverse the copy of the original tuple while the original tuple remains the same.

### Sorting a tuple:

#### Sort :

This is used to sort the original tuple.

#### Sorted:

This is used to sort the copy of the tuple while the original tuple remains the same.

Note: You can also use reversed and sort function together when you wanted to find the minimum or maximum in a tuple.

### Min/Max of the tuple

#### Min :

to find the minimum of the tuple.

#### Max :

to find the maximum of the tuple.

#### Sum :

to sum all the elements of the tuple.

### Tuple():

This is used to convert the given data type to tuple.

**Example:**

C:\Users\gs1-harinim\Downloads\Harini\_Study materials\tuple.py

# Sets:

## Properties of Sets:

1. Sets are enclosed within curly braces.
2. Sets are immutable.
3. Sets doesn’t allow duplicates.
4. Sets are always unordered.
5. Sets allow data of different data types.

**Note:** It allows you to declare a set with duplicate values but doesn’t return duplicate values.

## Accessing Sets:

1. You cannot access sets through the index.
2. Iterator is the only way of accessing,

## Built in functions/Methods:

### Adding Set:

1. **Add() –** Add a value to the set.
2. **Update()-**conversion and updates the value of a set.

### Remove Set:

1. **Remove() –** To remove an element from the set and throw exception if not present
2. **Discard() –** To remove an element from the set and will not throw exception if not present.
3. **Pop() –** randomly remove the element since set is unordered.
4. **Clear() –** clears the set completely.

### Copy():

Copy() is used to copy the set.

### Set functions:

1. **Union() –** used to join two sets.
2. **Intersection() –** used to return the common elements between two sets.

**Example :** C:\Users\gs1-harinim\Downloads\Harini\_Study materials\sets.py

# Dictionary:

## Properties of Dictionaries:

1. Dictionaries are enclosed within curly braces.
2. Dictionaries are mutable.
3. Dictionaries are ordered
4. Dictionaries does not allow duplicates.
5. Dictionaries also allows data of different types.

## Why Dictionaries allow duplicates?

Dictionaries does not allow duplicates because your keys cannot be multiple.

*Note: When you give multiple values to the same key, the latest value is considered and printed as output.*

## Accessing Dictionaries:

1. Accessing using keys.
2. Using get() – This returns the value of the key inside the get().
3. Using keys() – returns all the keys within the dictionary as list.
4. Using items()- returns all the values within the dictionaries as list.

**A screenshot of a computer code

Description automatically generated**

## Updates in Dictionaries:

1. Directly changing the value using key – You can pass the key on the left handside and assign the new value on the right hand side.

Example: dict\_value[“name”] =”Harini”

This will change the value in the “name” key to “Harini”

1. Update() – dict\_value.update({name: “harini”})

*Note: Though the field you are looking to update is not present in the dictionary, It will create a new key value pair instead of throwing an error.*

## Deletion in Dictionaries:

1. Pop() – argugments are must. Without arguments it will throw an error.

**Syntax: dictionary.pop(key)**

1. Popitem()- this will pop the last added value.
2. Clear() – empties the dictionary.

## Looping in Dictionaries:

1. **For loop – Accessing keys.**
   1. Anything that you try to access in the dictionary using a iterator, It access the key first as it is a address to locate the value inside the dictionary.

**Example:**

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

# Map and Lambda expression:

Map – Map is a one liner of for loop.

Lambda expression – Lambda is a one liner of for defining a functions

Filter – filters the list based on some conditions. It is a short from of for and if combination.

## Basic Lambda expressions: (https://onlinegdb.com/te13pGdUP)

A screen shot of a computer

Description automatically generated

This converts the string capitalized.

A close-up of a white background

Description automatically generated

This converts the string to upper case.

## Functions Versus Lambda:

The below code does addition and multiplication of three values.

A screenshot of a computer program

Description automatically generated

A computer screen shot of a function

Description automatically generated

## Conditional Lambda with AND OR:

A screen shot of a computer program

Description automatically generated

## Conditional Lambda with IF ELSE:

A screenshot of a computer program

Description automatically generated

## Conditional Lambda with IF ELIF:

The expression should be on the left-hand side of ‘IF’ for an IF statement and it should be on the right-hand side of ‘Else’ for an ‘ELSE’ statement.

A screen shot of a computer

Description automatically generated

## Map() in Lambda: (https://onlinegdb.com/DkzLHDoBP)

Map () in lambda is an aliter of loop. You use map function when you wanted to iterate through any iterable collection.

## Filter() in Lambda:

Filter() in lambda is an aliter of if condition and for loop. You use filter function when you want to give some condition and iterate based on that condition.

Examples for both:

A screen shot of a computer program

Description automatically generated

A computer screen shot of a black background

Description automatically generated

## List comprehension:

Syntax: expression for item in iteratable collection if condition

Example:

A screen shot of a computer program

Description automatically generated

# Map: (https://onlinegdb.com/qBGVr6aoZ)

* Any recursive action done for a iterable collection is achieved using map. To achieve this, Lambda, Filter etc. are used as supporting items.
* Map can be used within list because list is the only iterable collection which is mutable and ordered that can be accessed via indexing.

Syntax: map(function, iterable)

* + Function – name of the function(It can be user defined or Lambda)
  + Iterable – Iterable collection

Note:

1. Map is an object that is represented as 0x768bae560130, and it has no return type and returns the former always when printed.

Example:

A screenshot of a computer program

Description automatically generated

# Filter: (https://onlinegdb.com/swUBDFlGf)

* A combination of For loop and If condition is Filter. This is used when you wanted to set a condition in the function.

Syntax: Filter(function(condition),iterable collection)

* Function(condition) – Conditional executable.
* Iterable collection – Iterable collection such as ‘list’.

Example:

A screenshot of a computer program

Description automatically generated

# Scope:

Scope is the lifetime of a variable.

## Global Scope:

A variable or a functionality that is used across the program or a job and has distinct properties for each use case.

Example: Let us consider a project where the username is used in multiple places like logging into the account, order list for that account, Reward points for that account etc. Here, username has global scope as the same is used in many places in the program.

## Local Scope:

A variable or a functionality that is used in a specific place in a program or job with one unique functionality.

Example: Let us consider a project where the username is used in multiple places like logging into the account, order list for that account, Reward points for that account etc. Here, reward points have local scope only within that section.

Example:

A screenshot of a computer

Description automatically generated

# Module:

Module is nothing but a package which means importing another python file.

Example:

* Let us assume that there is a python file that you are currently working on where you wanted to perform certain mathematical calculations. Instead of defining them again and using it in a program, you can call another python file where these calculations are used and importing it inside the current file.
* Whenever you wanted to call any functions within the former file you can just import the same and use it in your code.

A .py file module that has basic arithmetic calculations in it:

A screen shot of a computer program

Description automatically generated

This module file is called in module excerside.py and the functionality is executed here:

A screen shot of a computer

Description automatically generated

Note:

* Whenever this import execution takes place, During the run time, there is a pycache file that gets created which has a list of all compiled files with extension .pyc in it which is shown below:

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

* This concept is applicable for all the built in functions or keywords used in python which indirectly implies that there is a set of .pyc files inside python which carries out the functionality that you call.
* .pyc file is a middle ware byte code file that is passed to the current file that you are trying to compile or the place you use it.

Program for importing built in libraries in Python:

## Using Import:

Using import, we import all the packages available inside math.

When you use ‘\*’, You are importing everything inside that package.

A computer screen shot of a program

Description automatically generated

## Using From:

Using from and import, we import only specific packages available inside math.

A computer screen shot of a program

Description automatically generated

## Accessing and locating a file from another location:

Import statements work only when the current file and the file you wanted to import is on same location. Incase , if it is in a different location, you must specify the path for the machine to locate and utilize the functions inside it.

A screenshot of a computer program

Description automatically generated

# OOPS in Python:

Object Oriented Programming Structure is known as OOPS. Java, Python , C++ and JavaScript are some of the examples of OOPS languages.

## Class:

A class is like an object constructor, or a “blueprint” for creating objects.

Note:

* A class should always start with class keyword.
* A class name should be titled.
* . operator is used to access the objects within a class and a class can have one to many objects.

## Constructor:

* Whenever an object is created for a class, Constructor is invoked automatically.
* Constructor is represented in Python as \_\_init()\_\_
* Constructor is the place where the validation is set.
* Constructor is used to define the subsequent actions towards a specific call.

Syntax: \_\_init\_\_(self,arg1,arg2)

* Self should always come first, and user can give any name.

Example:

1. Creating a class, Accessing object with and without constructor:

A computer screen shot of a program

Description automatically generated

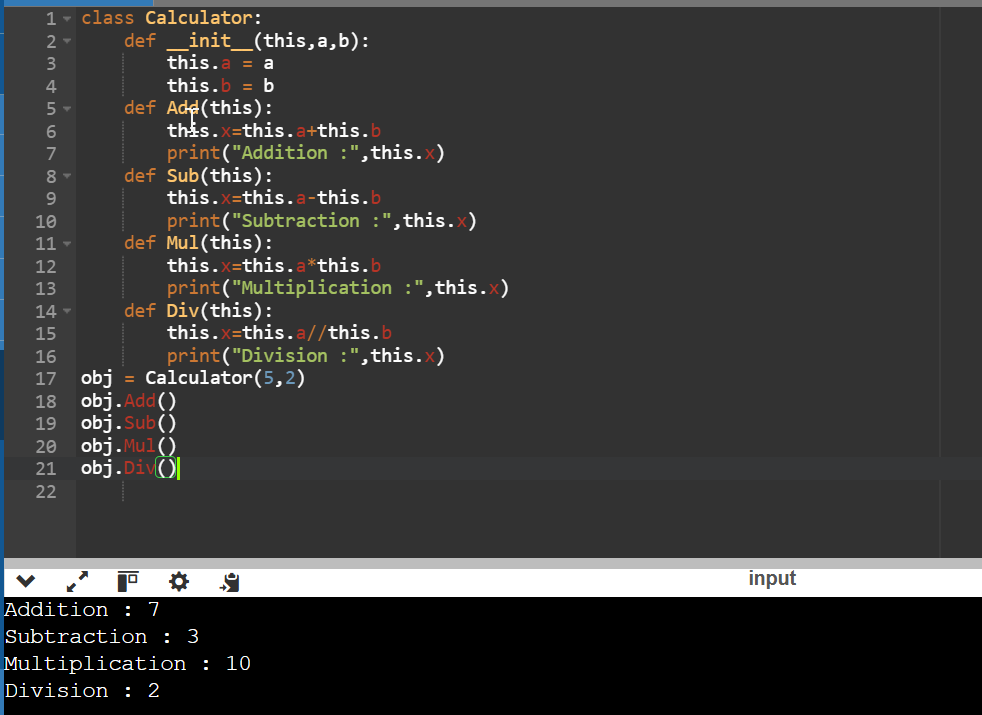
1. Converting the functions to a user desired data type: (You need not pass the arguments in this case as the object itself is converted to a desired data type)

A screen shot of a computer program

Description automatically generated

* Note: When you wanted to reinitialize the values assigned to the object in a class, You can just do the same by calling the object and equating with the value you want.
* Eg: o1.age =20 will reinitialize the age provided in the o1 object from 23 to 20.

**Example: Calculator program using Constructor and Class:**

****

## Inheritance:

Inheritance is the way in which the derived class inherits the properties of the base call. There are five different types of inheritance:

* Single level Inheritance.
* Multiple Inheritance.
* Multi-level Inheritance
* Hierarchical Inheritance
* Hybrid Inheritance.

### Single level Inheritance:

* A derived class inherits the properties from a single base class.
* There can be only one base class and one derived class.

Example:

#### Single level inheritance with Constructor:

A screen shot of a computer program

Description automatically generated

Output:

A black screen with yellow and green text

Description automatically generated

#### Single level inheritance without Constructor:

A screen shot of a computer code

Description automatically generated

Output:

A screen shot of a computer

Description automatically generated

### Multiple Inheritance:

* Multiple Inheritance is a type of inheritance in which the derived class inherits the properties of more than one base class.
* There can be many base classes but one derived class.

#### Multiple Inheritance with Constructor:

Example:

A screen shot of a computer program

Description automatically generated

Output:

A black screen with white text

Description automatically generated

#### Multiple Inheritance without Constructor:

Example:

A screenshot of a computer program

Description automatically generated

Output:

A screen shot of a computer program

Description automatically generated

### Multi-level Inheritance:

* Multi-level inheritance is a type of inheritance where the intermediate class inherits the properties of a base class, and a derived class inherits the properties of intermediate class.
* There can be only one base class and one derived class but can have many intermediate classes.

#### Multi-level Inheritance with Constructor:

Example:

A screenshot of a computer program

Description automatically generated

Output:

A screen shot of a computer

Description automatically generated

#### Multi-level Inheritance without Constructor:

**Example:**

A screenshot of a computer

Description automatically generated

**Output:**

A screenshot of a computer

Description automatically generated

### Hierarchical Inheritance:

Hierarchical inheritance is a type of inheritance where multiple derived class inherits the property of a single base class.

There can be only one base class but multiple derived class.

#### Hierarchical inheritance with Constructor:

Example:

A screenshot of a computer program

Description automatically generated

Output:

A black screen with white text

Description automatically generated

#### Hierarchical inheritance without Constructor:

Example:

A screen shot of a computer program

Description automatically generated

Output:

A screen shot of a computer

Description automatically generated

### Hybrid Inheritance:

Hybrid Inheritance is a combination of all the above inheritances.

Example:

A screen shot of a computer program

Description automatically generated

## Super():

Super function is used in cases where you wanted to call the constructor of the base class again and again. This super function is used to carry down all the functions of the base classes above.

You should give only the arguments as parameters for Super() and cannot give self.

Example:

A screenshot of a computer program

Description automatically generated

## **Polymorphism**:

* Polymorphism means “many forms”. The properties of one class can be used in many other classes.
* The same method name is used throughout the program.
* The properties are same throughout the program, but the use case varies from class to class.

Example:

In the below example, len() is used to find the length of the given iterable. This remains constant throughout any iterable used. Eg:list, string, tuple etc.

In the second example, the argument brand and model is applicable for all modes. But used separately as we wanted to define the expression inside the function uniquely.

A screen shot of a computer program

Description automatically generated

**Output:**

A screen shot of a computer

Description automatically generated

# **Formatting Springs:**

* String formatting is used to modify the modifiers based on certain requirements.
* Formatting string is always prefixed by “f” before the quotes.
* Modifiers are always enclosed within curly braces.

## **String literal:**

Combination of two data types:

* %s – String
* %d – Integer.
* %c – Character
* %f – float(e.g.:.%2f – with two decimal places)

# **Exception Handling:**

* Handling errors that occur during runtime is called as **Exception Handling.**
* When we have 1000 lines of code, The entire code will break and will throw an error if the line 2 of the code has an error. This makes the user to enable the execution from the start.
* To handle those errors and let the execution continue, we use **Exception Handling.**

**Syntax:**

try:

Piece of code that contains error

except:

Piece of code that catches the error.

Else: *\*optional*

Piece of code that gets executed after the try block when there are no errors.

Finally:  *\*optional*

Piece of code that gets executed no matter we have an error or not

### Try:

The try block lets you test a block of code for errors.

### Except:

The except block lets you handle the error.

### Else:

The else block lets you execute code when there is no error.

### Finally:

The finally block lets you execute code regardless of the result of the try- and except blocks.

## Common Errors:

* SyntaxError - interpretter error thrown if we have any compilation error.
* TypeError - datatype related errors.
* NameError - variable or function undefined/not found.
* IndexError -  index out of range error in iterables.
* KeyError - when dic does not contains that key.
* IOError - file handling related errors.
* ImportError - import package/lib/module not available in python.

## Examples:

A computer screen shot of a program code

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screen shot of a computer program

Description automatically generated

A computer screen shot of a program code

Description automatically generated

**Outputs:**

**A screenshot of a computer error

Description automatically generated**

# User Input:

Python allows for user input which means we can ask the user for input.

**Example:**

A computer screen shot of text

Description automatically generated

A screen shot of a computer code

Description automatically generated

**Output:**

A black screen with white text

Description automatically generated

# PIP:

**Commands:**

install Install packages.

download Download packages.

uninstall Uninstall packages.

freeze Output installed packages in requirements format.

list List installed packages.

show Show information about installed packages.

check Verify installed packages have compatible dependencies.

config Manage local and global configuration.

search Search PyPI for packages.

cache Inspect and manage pip's wheel cache.

index Inspect information available from package indexes.

wheel Build wheels from your requirements.

hash Compute hashes of package archives.

completion A helper command used for command completion.

debug Show information useful for debugging.

help Show help for commands.

General Options:

-h, --help Show help.

--debug Let unhandled exceptions propagate outside the main subroutine, instead of logging them to stderr.

--isolated Run pip in an isolated mode, ignoring environment variables and user configuration.

--require-virtualenv Allow pip to only run in a virtual environment; exit with an error otherwise.

-v, --verbose Give more output. Option is additive, and can be used up to 3 times.

-V, --version Show version and exit.

-q, --quiet Give less output. Option is additive, and can be used up to 3 times (corresponding to WARNING, ERROR, and CRITICAL logging levels).

--log <path> Path to a verbose appending log.

--no-input Disable prompting for input.

--proxy <proxy> Specify a proxy in the form [user:passwd@]proxy.server:port.

--retries <retries> Maximum number of retries each connection should attempt (default 5 times).

--timeout <sec> Set the socket timeout (default 15 seconds).

--exists-action <action> Default action when a path already exists: (s)witch, (i)gnore, (w)ipe, (b)ackup, (a)bort.

--trusted-host <hostname> Mark this host or host:port pair as trusted, even though it does not have valid or any HTTPS.

--cert <path> Path to PEM-encoded CA certificate bundle. If provided, overrides the default. See 'SSL Certificate Verification' in pip documentation for

more information.

--client-cert <path> Path to SSL client certificate, a single file containing the private key and the certificate in PEM format.

--cache-dir <dir> Store the cache data in <dir>.

--no-cache-dir Disable the cache.

--disable-pip-version-check

Don't periodically check PyPI to determine whether a new version of pip is available for download. Implied with --no-index.

--no-color Suppress colored output.

--no-python-version-warning

Silence deprecation warnings for upcoming unsupported Pythons.

--use-feature <feature> Enable new functionality, that may be backward incompatible.

--use-deprecated <feature> Enable deprecated functionality, that will be removed in the future.

**Example:**

A computer screen with text

Description automatically generated

A screenshot of a computer

Description automatically generated

A black background with white text

Description automatically generated

A black background with white text

Description automatically generated

A computer screen shot of a person

Description automatically generated

# File Handling:

* File handling is used to communicate with a machine.
* File 🡪Readable format 🡪 Operations to be performed in a file are defined 🡪 File is closed 🡪Compiler saves the file 🡪Goes to the interpreter 🡪 Output arrived.
* **CRUD operations** are always performed within a file.

## Open():

* Open() is used to open a file in some mode.

**Syntax: open(filename. Extension, mode/operation)**

### Modes/Operations in a file:

* r: open an existing file for a read operation.
* w: open an existing file for a write operation.  
      If the file already contains some data, then it will be overridden  
      but if the file is not present then it creates the file as well.
* a: open an existing file for append operation.  
      It won’t override existing data.
* r+: To read and write data into the file. This mode does not override the existing data, but you can modify the data starting from the beginning of the file.
* w+: To write and read data. It overwrites the previous file if one exists,  
      it will truncate the file to zero length or create a file  
      if it does not exist.
* a+: To append and read data from the file. It won’t override existing data.
* "x" - Create - Creates the specified file, returns an error if the file exists
  + In addition, you can specify if the file should be handled as binary or text mode
* "t" - Text - Default value. Text mode
* "b" - Binary - Binary mode (e.g. images)(all system defined reading is used in binary mode)

**Syntax 2:** open(filename.ext, operation/mode\_combined\_with\_file type(text/binary))  
eg - open(hello.txt, rt) or open(hell.txt,rb)

### **Read – Write – Append:**

1. Read – Reads the file.
2. Write – Writes the file. The data gets overridden completely. It creates a file if not present.
3. Append– It appends something to the existing content without overriding the existing content.

A screenshot of a computer program

Description automatically generated

### With statement:

1. There is no need to call file.Close() when using with statement.
2. It handles the exceptions.
3. It reads the file automatically.

A screen shot of a computer

Description automatically generated

A screenshot of a computer program

Description automatically generated

**Output:**

A screenshot of a computer program

Description automatically generated

A black background with white lines

Description automatically generated

## Remove/Delete a file in Python:

* Import OS.
* Use remove()

# RegEx:

* Python has a built-in package called re, which can be used to work with Regular Expressions.
* Import the **re** module
* RegEx is a sequence of characters that forms a search pattern.
* RegEx can be used to check if a string contains the specified search pattern.

## Email address Validator:

A screenshot of a computer program

Description automatically generated

## Meta Characters:

|  |  |
| --- | --- |
| ^ | Starts with |
| [] | Range |
| + | concatenation |
| .\ | Anything can come in between. |
| {} | Exactly the specified number of occurrences |
| | | Either or |
| () | Capture and group |
| \* | Zero or more occurrences |

# **Libraries in Python:**

## Openpyxl:

* This is a library which is used to access an excel workbook.
* There are other libraries like **pandas, xlrd, xlswriter, iornxl** which is used to access a workbook apart from openpyxl.

### Modules in Python:

#### Read:

Please refer to the screenshot below.

**Example:**

A screen shot of a computer program

Description automatically generated

**Output:**

A screenshot of a computer

Description automatically generated

#### **Write:**

* This will override the existing information instead of appending to the existing information.
* You can create a new work book by using workbook()

**Example:**

**A screenshot of a computer program

Description automatically generated**

**Output:**

**A screenshot of a spreadsheet

Description automatically generated**

#### Append:

* You can append the new data to the existing information by using load workbook method.
* This method is used to invoke the existing workbook.

**Example:**

A screenshot of a computer program

Description automatically generated

**Output:**

A screenshot of a computer

Description automatically generated

## **Pandas:**

* Pandas can do the following operations:

A cartoon panda sitting in a chair with a pencil and a book

Description automatically generated

* It can also compress and decompress a zip file.
* We can access SQL files, HTML files, JSON files, Excel and CSV files in Pandas.

### **Read\_Excel:**

1. Read\_Excel function takes care of all the functionalities done in openpyxl by loadworkbook(),active(),cell(),value().

# **Machine Learning:**

* Machine learning is nothing, but you train a model to bring a desired output.
* Machine Learning is the top of Artificial Intelligence.
* Super computing is also a part of Machine Learning that gives the result as Boolean value.
* Quantum computing is also like Super computing that takes all the possibilities apart from the usual situations.

**E.g.: When you toss a coin, Heads and Tails are the usual probability. Quantum computing goes a extra mile in analyzing what happens if the coin stands in between.**

# **Deep Learning:**

* Deep Learning is an extension of Machine Learning.

# **Data Roles Map:**

**Data Engineer 🡪 Data Analyst 🡪 Data Scientist**

* **Data Engineer** – Data Engineer is used to provide all the data required by data analyst and data scientist for analysis and forecasting.
* **Data Analyst** – Analyses the data into various sections based on the current trend and past trends.
* **Data Scientist** – Forecasts the future with current trends and past trends.

# **Tensor Flow:**

* Tensor flow helps you to build your model.

# **OpenCV:**

* This is a library to classify images.
* Some of the examples are tracking CCTV images by tracing or boxing moving objects.

**Example:**

When you wanted to track a CCTV footage, It boxes around the moving objects and then takes and feeds the images to AI to determine the features of the moving objects such as gender, color etc.

# **NLP:**

NLP is nothing but Natural Language processing to determine the result based on sound which is in turn converted into data frames.

Eg: Kannum Kannum Kollai adithaal locker scene.

# **Matplotlib:**

* Matplotlib is the mathematical plotting library which is used for graphical representation of data.
* There are numerous functions to access every corner of the graph including labels, grids, values etc.

**A diagram of a graph

Description automatically generated with medium confidence**

**Example:**

**A screenshot of a computer program

Description automatically generated**

**Output:**

**A graph with a line

Description automatically generated**

# Seaborn:

Sea born is used to represent the data in graphical form and produce more accurate results.

## Boxplot:

Box plot is used to represent the data in graphical form where there is two comparisons involved.