Basics of python

This document contains the information and knowledge required to understand python language and its basic features. With this document a fresh learner can start understanding this machine language.

Exploring Machine learning

Table of Contents

[Why Python 2](#_Toc494897141)

[Some popular libraries in Python 2](#_Toc494897142)

[Basic mathematical functions with symbol 2](#_Toc494897143)

[Variables in Python 2](#_Toc494897144)

[Datatype in Python 3](#_Toc494897145)

[Type casting and basic operation 6](#_Toc494897146)

[Data structure in Python 9](#_Toc494897147)

[Lists & Tuples I in Python 10](#_Toc494897148)

[Sets 13](#_Toc494897149)

[Dictionaries 14](#_Toc494897150)

[Arrays in Python 15](#_Toc494897151)

[Matrices in Python 17](#_Toc494897152)

[Data Frames in Python 17](#_Toc494897153)

[String operation in Python 18](#_Toc494897154)

[Date and Time Value Manipulation 19](#_Toc494897155)

[Regular Expression 19](#_Toc494897156)

[Conditions in python 20](#_Toc494897157)

[Loops in Python 20](#_Toc494897158)

[Functions in Python 21](#_Toc494897159)

[Object and Classes 22](#_Toc494897160)

[Error and code formatting in Python 23](#_Toc494897161)

[Python File Modes 24](#_Toc494897162)

[Reading CSV, Excel, and built in datasets with in Python 25](#_Toc494897163)

[Reading text file in Python 26](#_Toc494897164)

[Writing and saving File in Python 26](#_Toc494897165)

**PYTHON 101**

## Why Python

1. Python is a high level open source programming language.
2. It is a powerful, fast and dynamic programming language that runs everywhere.
3. It is interactive, object oriented, and very easy to learn.
4. It is very useful for web development, and scientific computing. With its vast libraries, this language is useful for data visualization and data analytics.
5. Easy to read, allow rapid development, rich text and list support, lots of third party libraries.

Note: Semicolon ; is used to separate multiple statement on the same line. Example below

Print “Hello” ; Print “World”

Output : Hello World

## Some popular libraries in Python

1. Numpy- Provides advanced math functions.
2. Pandas- Data analysis libraries; offers data structure and operations for manipulating numerical tables and time series.
3. Scipy- A library of algorithm and mathematical tools.
4. Ipython- With completion, history, shell capabilities, and a lot more.
5. Matplotlib- A numerical plotting library, useful for data scientist or data analyst.
6. SKleargn- It’s also known as Scikit learn. Used for machine learning and data mining in python.

## Basic mathematical functions with symbol

Addition +

Subtraction –

Multiplication \*

Division /

Remainder or Modulus %

Exponents \*\*

Division without decimal //

## Variables in Python

They are container which store a value. Variable name can be anything and they can hold any data. String, character, number, decimal.

5 rules to create a variable name

1. Variable name must start with a letter or \_ (underscore) followed by letter, number or underscore.
2. After first letter it can be number, underscore, letter.
3. Variable name is case sensitive, so each time it should be similar.
4. Try to avoid capital I or O because they can be misinterpreted for 0 or 1.
5. There are few reserved words or keywords in python so better do not try to create a variable name for those reserved words.

*STRINGS ARE A SEQUENCE OF CHARACTERS.*

Anything that is inside a pair of quotations, either single or double, is considered a string.

Be careful when adding multiple string. Example below.

Print “Welcome” + “to India”

Result : Welcometo India

As shown above its mandatory to provide space explicitly.

Print “Welcome “ + “to India” or Print “Welcome” + “ to India”

Result : Welcome to India

Python automatically adds a space between the strings when using comas. Example below.

Print “Welcome” , “to India”

Result : Welcome to India

## Datatype in Python

Below are the common datatypes in Python.

* Integer
* Float
* String
* Boolean
* Long
* Complex
* Datetime

**Number:**

There are 2 types of number in Python :

1. Integer- These are whole number, negative or positive but always without decimals.
2. Float- These are numbers with decimal point

**String:**

We can write string using either single or double quotes. Number stored as string cannot be used to perform mathematical operation.

Ex: String\_Var “10 is highest”

Print String\_Var

Result : 10 is highest.

String\_Var “177”

Var\_Store String\_Var + 10

Result : 17710

**Boolean:**

Boolean values are 2 constant objects. True and False. Boolean variable holds either true or false value.

True and False must be written with capital first letter and no quotation single or double. Only that way Python recognize it as Boolean.

Note : Boolean are sub type of plain integer.

**Distinct numeric type:**

There are 4 distinct Numeric type.

1. Plain integer (B) Long integer (C) Floating point number (D) Complex number

|  |  |  |  |
| --- | --- | --- | --- |
| Plain Integer | Long Integer | Floating Number | Complex Number |
| These number have 32- bit of precision. | They have unlimited precision. |  |  |
|  |  |  |  |

**Long datatype:**

These data types hold long integer. They are only available in version Python 2.

Max value python can hold in integer is 2 to the power of 63 minus 1

**Complex datatype:**

These holds real and imaginary complex numbers such as square root of negative 1 or i. A complex number is expressed in form of a + bi, where a and b are real number and i is imaginary unit that satisfy the equation i squared = negative 1.

In python we can put “j” or “J” after a number to make it imaginary. Example below.

1j, 1J

**Date time Datatype:**

In python, various modules are present.

1. Time module – It provide time access and conversion.
2. Calendar module – It has general calendar related functions.

To use data time data type, we need to import it. Example below.

Import datetime

length\_of \_album = datetime.time (0,42,19)

print length\_of\_album

print ‘Hour :’, length\_of\_album.hour

print ‘Minute :’, length\_of\_album.minute

print ‘Second :’, length\_of\_album.second

print ‘Microsecond :’, length\_of\_album.microsecond

print ‘TimeZoneInfo :’, length\_of\_album.tzinfo

00:42:19

Hour : 0

Minute : 42

Second : 19

Microsecond : 0

TimeZoneInfo : None

## Type casting and basic operation

The data type of a value can be change. Ex- changing an integer to string. This is called type casting.

Type casting can be done using shorthand like below

Integer “int”

Boolean “bool”

Float “float”

String “str”

Example code to apply type casting

Artist\_age = “25”

Artist\_age= int (Artist\_age)

Artist\_age

Note : It’s not possible to convert a string into number which contain at least a character including space. Type cast is only possible to convert string into number when string is a whole number like “10”.

A string value in form of decimal cannot also get converted into integer. Like int (“13.7”) is not possible to get converted into integer.

Some special case:

1. When convert a whole integer into float. Decimal value is added by default.

Ex- float(12)

Result 12.0

1. When converting a number with decimal into float. Decimal value retains.

Ex- float(14.4)

Result 14.4

1. When converting a float value to integer. We lo0se decimal part.

Ex- int (12.432)

Result 12

1. When converting Boolean. Any numerical value that is not zero and any string which is not empty including a space will return TRUE.

Ex- bool(), bool(12), bool(0), bool(“Hello”), bool(‘hi’), bool( ), bool(‘ ‘), bool(“ “), bool(“”)

Result False, True, False, True, True, False, True, True, False

**Operations on string**

* Concatenation- It combines two or more strings into a new string object.

Example code below-

First\_name= “Manjeet”

Second\_name= “Singh

Full\_name= First\_name + “ “ + Second\_name

Print Full\_name

Result Manjeet Singh

* Variables of same type can be like integer can be added, subtracted, multiply, divide etc.

Example below code

Num1 = 10

Num2 = 20

Num3= Num1+Num2

Print Num3

Result 30

* We cannot add a string and an integer. But we can turn a number into string using str () function.

**Operator precedence –** It determines the order in which operations are processed.

Below is the order of precedence from highest to lowest

()

\*\*

\*

/

+

-

Some more example

Num1 = 10

Num2 = 20

Num3= Num1+Num2

Print Num3 \* 2

Result 60

First\_name= “Manjeet”

Second\_name= “Singh

Full\_name= First\_name + “ “ + Second\_name

Print Full\_name \* 3

Result Manjeet Singh Manjeet Singh Manjeet Singh

*It means + means concatenation and \* means multiple concatenation*

To give comment in python code use #

Example code

Str\_Var = “Manjeet” # this is name store in Str\_Var

To create an array of string. See example below

In [ ] :

String\_Array\_Create = [“Manjeet”, “Singh”, “panwar”]

## Data structure in Python

Data structure is a way to organize data like arrays.

There are 4 types of data structure in python

1. Strings
2. Lists
3. Tuples
4. Dictionaries (or hash, maps)
5. **Strings**

* Example – String1= “Hey there can you hear me”
* Can use single or double quotes.
* Dynamically typed
* Strings are immutable
* Strings indexing

Example – String2= “Hello”

Print String2[0] Result H

Print String [1] Result e

Print String [1:3] Result el

Print String [1:] Result ello

Print String [:4] Result Hell

Few more functions in strings

* s.lower(), s.upper() -- returns the lowercase or uppercase version of the string
* s.strip() -- returns a string with whitespace removed from the start and end
* s.isalpha()/s.isdigit()/s.isspace()... -- tests if all the string chars are in the various character classes
* s.startswith('other'), s.endswith('other') -- tests if the string starts or ends with the given other string
* s.find('other') -- searches for the given other string (not a regular expression) within s, and returns the first index where it begins or -1 if not found
* s.replace('old', 'new') -- returns a string where all occurrences of 'old' have been replaced by 'new'
* s.split('delim') -- returns a list of substrings separated by the given delimiter. The delimiter is not a regular expression, it's just text. 'aaa,bbb,ccc'.split(',') -> ['aaa', 'bbb', 'ccc']. As a convenient special case s.split() (with no arguments) splits on all whitespace chars.
* s.join(list) -- opposite of split(), joins the elements in the given list together using the string as the delimiter. e.g. '---'.join(['aaa', 'bbb', 'ccc']) -> aaa---bbb---ccc

1. **Lists**

* Example- My\_List = [1,”Hi”,3]
* Addition (Concatenation), indexing, slicing, and length calculation like string.
* Lists are mutable.
* Operations involving lists:

My\_list[2] = “Hello”

My\_list.append(“Good Afternoon!”)

Del My\_list[1] # Deleting the indexed item in the list

My\_list.pop() # Removing and returning the last item in the list

My\_list.pop(2) #pops the 3rd item (index 2) from the list

## Lists & Tuples I in Python

List and tuples both are ordered it means every element will have index.

Lists and tuples are called compound datatypes and are key types of data structure in Python.

List can hold any type of data similar or different datatypes. Each element of the sequence is assigned a number that indicates its position or index. Lists are Mutable, it means their data can be changed.

Index starts with 0 and go on till the last element of the list.

Append () is used to add an element to a list. Only one element is added in list that means only an index is appended.

Extend () is used to add element in a list. With Extend we can add as many elements all elements will have unique index.

Del() is used to delete an element from list.

Del(List1[0])

With above command element at index 0 will be deleted.

Split is used to convert string into a list. Every space becomes the separator like example below.

“Hard Rock Café”.split()

[“Hard”,”Rock”,”Café”]

We can also specify a delimiter with in split function to separate string and create a list.

“A,B,C,D”.Split(“,”)

[“A”,”B”,”C”,”D”]

We can also create alias of list.

A=[“Hard rock”, 10, 1.4]

B=A

Now A and B are referencing same list. If we change any item in A or B corresponding index value or element will be modified.

Multiple names referencing the same object is known as aliasing.

B[0]= “Hard rock”

A[0]=”Banana”

Now List A and B will have Banana as index 0 in the elements.

If we want separate list we need to make a clone of list.

A=[“Hard rock”, 10, 1.4]

B=A[:]

Now variable A reference List1 and variable B referencing another List which is a copy of List1 only.

Anytime use help(<VARIABLE NAME>) to seek some guidance or assistance.

Tuples are like list, but the data cannot be changed, it means they are immutable. Tuples elements are also accessed using indexing.

Note:

* Tuples are fixed size in nature whereas list is dynamic.
* Tuples are faster than list.
* Use Tuples when we have constant list of values.
* Some tuples can be used as dictionary keys whereas list can never be used as dictionary keys.
* List use [] whereas tuples use () to store elements.
* Ex: album\_list = [“Michael Jackson”,”Thriller”,1982,(“Pop”,”Rock”),32,45]

As shown above one of the element is tuple in list.

Ex: album\_tuple = (“pop”,”rock”,”progressive”,R&B)

To know the length of elements in list or tuple use length function like

Len (album\_list)

return 6

len (album\_tuple)

return 4

To Find the minimum or maximum value in a list use min and max function

Min (album\_list)

Return 32

Max(album\_list)

Return Thriller

Thriller is returned in max function because integers comes before characters.

To Find the minimum or maximum value in a tuple use min and max function

Min (album\_tuple)

Return R&B

Max (album\_tuple)

Return rock

**Imp- upper case characters are sorted before lowercase character.**

Python is a zero-index language. So, if you have a list of albums, they would be index as 0,1,2 and so on. Similar way if you have list of tuples they would also be index as 0,1,2 and so on.

album\_list.index(“Michael Jackson”)

return 0

album\_list.index(32)

return 4

In case you want to sort the elements in list then use album\_list.sort(). Tuples cannot be sorted.

Once sort operation is complete indexing will also change according to elements. Also by default sort is in ascending order we must explicitly code to make elements appear in descending order. Order of sorting is below.

1. First Numbers
2. Second Sets. Within sets if there are multiple elements those will sorted out.
3. Third Numeric String
4. Last Characters strings, with uppercase character before lowercase.

To sort a list in descending order user album\_list.sort(reverse=true)

To reverse the elements in a list user reverse method it’s not like sort. It will just reverse the elements which means first element becomes last and last becomes first and so on. Each element in list will be reversed and position/index will change.

album\_list.reverse()

To convert a tuple into a list user below method or function.

album\_tuple = (“pop”,”rock”,”progressive”,R&B)

Now this is a tuple with some elements.

album\_list2-list(album\_tuple)

album\_list2

return [“pop”,

“rock”,

“progressive”,

“R&B”]

As you can see above all elements appear in album\_list2 and output also visible in [] which shows tuples are converted in another new list. Now it’s easier to sort elements of tuples because we have made another list 2 to get result sorted out.

**IMP- Functionally we cannot sort elements of tuples but if we convert elements of tuples in list then we can apply sort on new list.**

## Sets

* Sets are type of collection.
* Like list and tuples we can input different Python types.
* Unlike lists and tuples they are unordered, it means sets do not record element position.
* Sets only have unique elements it means an element can appear only once.
* To define a set we use { } like Set1={“Pop”, “Rock”}
* We can convert a list into set by using function set. This is called type casting. Basically this is done to remove duplicate items in list.

List1= [“Michael Jackson”, 10, 19, 10]

A\_set=set(List1)

A\_set : {“Michael Jackson”, 10,19}

To add an element in set. Consider we have a set named as A

A.add(“NYSE”)

Note : If we add same element twice then no operation takes place and no change done to set as we cannot have duplicate element in set.

To remove an element from set A

A.remove(“NYSE)

In operator is used to look with in set element if an element is found then return true else false.

& is used to find common elements in sets. Example below.

Album\_set1= {“Hard”,19,71,32,31.4}

Album\_set2={”The”,19,54,60}

Album\_set3=Album\_set1 & Album\_set2

Result Album\_set3: {19}

Union can also be done for multiple set to combine all elements of sets. Example below

Album\_set1= {“Hard”,19,71,32,31.4}

Album\_set2={”The”,19,54,60}

Album\_set3=Album\_set1.union(Album\_set2)

Issubset method is used to find whether a set is a subset of another set.

Album\_set1= {“Hard”,19,71,32,31.4}

Album\_set2={19,32}

Album\_set2.issubset(Album\_set1)

TRUE

As the AlbumSet2 is subset of Album\_set1 true is returned.

## Dictionaries

Dictionaries are a type of collection in Python. In list we have index and elements but in Dictionaries we have Key and Value. Key can be thought of as index and mostly they are character not integer as in list and value can be thought of as element in list.

* Dictionaries are denoted with {}
* The keys must be immutable and unique.
* The values can be immutable, mutable and duplicates
* Each key and value pair is separated by a comma

Below is an example of creating a dictionary

Dict={“Thriller”: 1980, “Bodyguard”: 2000}

Key is used to look at the value like below.

Dict[“Thriller”]:

Output is 1980

To add an element or value in Dict use below code.

Dict[‘Graduation’]= ‘2007’

Dict[“Grade”]= ‘A’

Above 2 commands will add 2 values and 2 keys in already defined Dict dictionary.

To delete an entry from dictionary.

del(Dict[‘Thriller’])

Above code will remove key and value for Thriller

To find or look up in a dictionary use in command

‘Thriller’ in Dict

Returns true if key found in dictionary or else false.

To see all keys exist in a dictionary use below command.

Dict.Keys()

Output will be all keys available in Dict dictionary in square bracket [].

Similar way we can see all elements in a dictionary.

Dict.values()

Output will be all values in the Dict dictionary in square bracket [].

Dict() is used to convert a tuple of element into a dictionary.

Example dict(a) considering a as tuple having element as pair of key and value.

## Arrays in Python

* An array is structure containing data of same type. It can be string, character or integer.
* Arrays can be multi dimension it means they can display data in rows and columns.
* To create an array, we first create vector i.e. values with in square bracket [ ].
* After vector, we create matrix using variables for rows and columns.

Example code below.

Album\_vector = [“HardRock”, “Silly”, “Emotional”, “Rock”]

Album\_vector

[“HardRock”,

“Silly”,

“Emotional”,

“Rock”]

ncols=2

mrows=2

matrix= [ [0] \* ncols for i in range (nrows)]

matrix [0] [0] = “Hardrock”

matrix [0] [1] = “Silly”

matrix [1] [0] = “Emotional”

matrix [1] [1] = “Rock”

matrix

Result will be a 2\*2 table is displayed.

Note -> We can also create an array using Numpy. Advantage of using numpy is we do not need to create matrix as created in above code.

Import numpy as np

Album\_array= np.array(album\_vector)

Album\_array

Result will be a table 2\*2 displayed.

To access an element in array we need to provide row no and column no at same time. Example code below.

Album\_array [2,1]

Above code will fetch element of row 2nd and column 1st.

We can also fetch or extract subset from an array using range : (colon) example code below.

Subset\_matrix= Album\_array[2:4]

Here 2 denote row number, it will be 3rd row because index starts from 0 and 4 denote no of element.

To get shape and datatype of array use dtype

Album\_array.shape

Result is (2,2)

Album\_array.dtype

Result dtype(‘531’)

## Matrices in Python

Matrices are like arrays with the condition that they should be 2-dimensional.

A matric can be formed using the numpy matrix()

Import numpy

Numpy.matrix([1,2,3],[4,5,6],[7,8,9])

By default a matrix is formed column wise.

## Data Frames in Python

* A data frame is a structure that contains a group of correlated information.
* It is same as list but each variable has a vector of elements of the same type.
* We use pandas function to create data frames.

Import pandas as pd

Df= pd.DataFrame(music\_dataset\_dict=[‘artist’,’album’,’year released’)

To access data in data frame use below codes. A data frame is stored as table and can be displayed in table as well using python.

Df[0:0]

Above code will return the name of all columns in data frame.

Df[3:4]

Above code will return 3rd index row which starts from 0. So 4th row of data frame is returned with all columns.

Df[:3]

Above code will return first 3 rows of data frame.

Df[5:]

Above code will return all starting from index 5.

Df(‘album’)

Above code will return all values in column album. This method is used to return all elements of a column.

Similar way addition, subtraction, mean or other calculation can be easily applied on data frame.

Using head or tail function

Import pandas as pd

Df.head(1)

Above code return the 1row of data frame. In case (1) is not specified then first 5 rows are returned.

Import pandas as pd

Df.tail(2)

Above code will return last 2 rows of data frame.

Now it’s time to create a small data frame.

Data= {‘S.no’:[1,2,3], ‘Name’: [“Maan”,”Kamal”,”Jai”]}

Df=pd.DataFrame(data)

Df

Result will be 2 column S.no and Name with 3 rows.

Now add a new column to data frame

Df[“Class”]= “V”,”VI”,”X”

Df

Now to add a new row in data frame, use append function.

Df.append({‘S.no’:4,’Name’:”Bum”,’Class’:”IX”},ignore\_index=True)

To delete a row from data frame, use drop function.

Df.drop(2)

Above code will remove 2nd index which would be 3rd row as counting of index starts from 0.

To delete a column from data frame.

Df= df.drop(“Class”,1)

## String operation in Python

* In python a string is a sequence of character.
* We can create string from single or double quotes.
* A string can be spaces or digit.
* A string can also be special character.
* A string is stored as an array of indexes.

If a string variable is Str\_V1=”Jai”

Then it will have 0,1,2 as index.

We can refer or access these indexes like Str\_V1[1]

We can also use -ve indexing with string. The last character starts with -1

So i =-1 a=-2 j=-3

Str\_V1[-2]

Result is a

Len is used to calculate the length of string.

len(Str\_V1)

Result is 3

We can concatenate a string using +.

We can multiply the string using \* to repeat the same string

We cannot edit the string but we can add element is string using concatenate operation.

\n is used to enter to new line

\t is used to provide tab between element in string.

In case we want to display \ in a string then use \\ (twice) to display \ once.

We can find, replace, upper, lower case the string element.

## Date and Time Value Manipulation

* In python we can time parsing, formatting and arithmetic.
* Time values are represented with the time class. Times have attributes for hour, minute, seconds, microseconds. It can also include time zone information.
* A time instance only hold time and not the date associated with the time.
* Calendar date values are represented with the date class. Its Instances have attributes for year, month, and day.
* It always follows the Gregorian calendar.

## Regular Expression

These are special type of expression in python. Used to find or match patterns in string or text.

Suppose we want to find email address from list of email addresses available. We can start with below code.

.+@.+

. Means matches with character.

+ Matches the preceding pattern element one more time. So a . and + represent any possible combination of characters possible.

@ Means this should exist. In an email address we would have @ symbol.

Regular expression is used mostly in data cleaning and mining. Even used for text parsing during code compilation.

## Conditions in python

Conditionals are simple yes/no questions that compare a variable’s value to something that program is looking for.

Sample if else block.

Album\_year = 1970

If Album\_year> 1980:

Print “Album year greater than 1980”

Else:

Print “Album year less than 1980”

Sample elif block. In elif block we have another condition to be tested. In case IF statement is true ELIF will be skipped.

Year = 1970

If (Year < 1980):

Print “Too Old”

Elif Year >=1988:

Print “Not so old”

## Loops in Python

We need to first understand the range () before jumping to loops.

Range () starts with 0. If a positive number is passed as argument then it output will be sequence.

Range (3)

Output [0,1,2]

Range (10,15)

Output [10,11,12,13,14]. Here first argument is smaller than second argument so a sequence is generated with one less than the second argument.

Loops perform a task over and over. Example of For loop below.

Squares = [“Red”, “Yellow”, “Green”]

For i in range(0,3):

Squares [i]=”White”

While loop is like for loop. But instead of executing a statement a set number of times, a while loop will only run if a condition is met. Example code below.

Squares = [“Orange”, “Orange”, “Purple”]

Newsquare=[]

i=0

while (square[i]==”Orange”):

Newsquare.append(square[i])

I=i+1

## Functions in Python

A function is a re-useable block of code which perform operation specified in it. There are 2 types of functions. We can also return value from a function and can also use if-else or any other function.

1. User defined functions- These are custom functions created and defined by users. See example code below.

<Def> <Function Name> (Parameter 1, Parameter 2, Parameter N) :

* Def is a keyword which means define
* Function name is any name which user wants to give to function. It’s best to give name as per the operation performed by it.
* Parameter are values need to be passed or variables used in functions.

1. Pre-defined function- These are already defined in python or available in library Example sum(), len(), max() etc.

## Object and Classes

* Python has lots of data types.
* Int, String, float, list, dictionaries, Boolean.
* A class contains data attributes and methods.

In python Each is an object.

* Every object has a type.
* An internal data representation (a blueprint)
* A set of procedures for interacting with the object (methods).

To find a type of any object use type () function like below

Type (“ABC”)

<Class str>

Type(5)

<Class Int>

Define class in python.

Example below.

Class Circle

Its attributes can color, radius

Class Rectangle

Its attributes can be color, height, width

Class Circle (object):

* Class is a keyword use to define a new class by user.
* Circle is the name of the class
* Object refers to the parent of the class.
* \_init\_ is the constructor used to initialize data attributes.

Class Circle (object):

Def \_init\_(Self, radius, color):

Self.radius=radius;

Self.color=color;

RedCircle=Circle(10,”red”) # This is creating object of class.

Class Rectangle (object):

Def \_init\_(Self, height, width, color):

Self.height= height;

Self.width= width;

Self.color= color;

Black=Rectangle(2,5,”Black”)

Methods in Class- We can create function or methods with in class to do extra operations. Suppose we want to increase the radius of circle. To achieve this task, we can create a method. Example below.

Def add\_radius(Self, r):

Self.radius=Self.radius +r

C1.add\_radius(8) # Calling method.

Note : dir (Nameofobject) is used to obtain the list of attributes and methods associated with the class.

## Error and code formatting in Python

There are 2 types of error in programming.

1. Syntax error- These are also known as parsing errors. Mostly typo errors in code.
2. Exceptions- Errors detected during execution of code are called exceptions.

Note: # is used to give single line comment with in the code. And “”” triple quotation is used to give multiline comments with in python code. Just add “”” This is a comment “””

\n used to display string or object to new line.

In case \ need to print or display use it twice. \\ this will print \ once.

## Python File Modes

Python supports reading and writing into many different file formats.

JSON, XML, CSV, TXT, PDF, HTML, RDF, TIFF, JPEG

Example of file input:

* Reading user preferences
* Importing data.

Example of file output:

* Outputting log files for debugging
* Saving user preferences
* Exporting data

Opening a file in python

myFile= open (“FileName.txt”)

OR

myFile= open (“C://User/FileName.txt”)

We can also open file in read or write mode.

myFile= open (“FileName.txt”, “Mode (s)”)

s denotes the mode. There are several modes in python r, w, a, x, t, b, +

Always it’s a good practice to close the file once job or operation is done.

myFile.close() # will close the file.

Try/Finally block in python

Try:

myFile= open (“FileName.txt”, “w”)

finally:

myFile.close()

myFile.write(string)

Mode W- Write the content to file, create new if file do not exist. Truncate all content if file exist.

Mode A- Append additional content to file, create new if file do not exist. Do not Truncate content of file.

Mode X- Creates a file, return error if file exist.

Another way to write into file.

myFile= open(“FileName.txt”, “W”)

myFile.WriteLines(“The top 3 resources are\n Thriller\n Rock \n Common”)

myFile.close

Python read the content of file through virtual cursor. Tell and Seek methods are used. Tell is used to obtain the position of virtual cursor.

# myFile.tell()

Print myFile.tell()

^

16

The seek method is used to change the position of virtual cursor.

# myFile.seek([offset, position])

myFile.seek(30)

^

30

## Reading CSV, Excel, and built in datasets with in Python

Python is a popular language for text and file munging due to :

* Simple syntax for interacting with files.
* Intuitive data structure
* Convenient features: tuple packing and unpacking

Read a CSV file – It’s a comma separated values file. Contains structured data in table form.

Import pandas

Df= pandas.read\_csv(“path of the file”)

Read an excel file with xls, xlsx extension. XLRD module or package is required to read excel file in python. If it’s not available then install it using !pip install xlrd

Import pandas

Df= pandas.read\_excel(“path of the file”)

Whenever we read tabular data in python, the default method of structuring this data is using data frames.

Data frames are extremely versatile and python presents many options to manipulate them.

Following ways are used to retrieve information from excel or csv file.

Import pandas

Df= pandas.read\_csv(“path of the file”)

To access a column in csv use Df[“ColumnName”]

To access specific row, we use position or name using a couple of methods.

Data= Df.iloc(2)

Data=Df.ix[1]

To access a cell, we use Data= Df.iat[0,0] or Data= Df.at[0,Column]

## Reading text file in Python

Default mode is read mode. If a mode is not specified always file is open in read mode.

myFile= open (“”File.txt”)

myFile.read()

There are several methods used in reading text file.

## Writing and saving File in Python

We can export the content of file and save in local hard drive. Below example for text file.

Import numpy as np

Matrix= np.arrange(10). Reshape((2,2))

Matrix

Array([[1, 3],

[5, 8]])

np.savetxt(“Specify Path”, Matrix)

How to export data in CSV.

Import pandas as pd

Matrix = {‘0’ : [1,2],

‘1’ : [3,4]}

Matrix

Df= pd.DataFrame(Matrix, columns= [‘0’,’1’])

Df

How to export data in excel.

Import pandas as pd

# Create a Pandas dataframe from same data.

Matrix = {‘0’ : [1,2],

‘1’ : [3,4]}

Matrix

Df= pd.DataFrame(Matrix, columns= [‘0’,’1’])

Df

# Create a pandas excel writer XLSX

Writer= pd.excel(writer, sheet\_name=”Sheet1”)

# Close the pandas excel writer and output the excel file.

Writer.save()

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* END OF BASIC COURSE \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*