INTERNSHIP PROGRESS REPORT

Submitted in the partial fulfilment for the award of the degree of

BACHELOR OF ENGINEERING

IN

INTERNET OF THINGS

Submitted by:
Rishabh Anand
19BCS4525

AT
HIGHRADIUS

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

APEX INSTITUE OF TECHNOLOGY

CHANDIGARH UNIVERSITY, GHARUAN,

MOHALI - 140413, PUNJAB

MARCH 2022

Student Name	Rishabh Anand
Student UID	19BCS4525
Student Email (@cuchd.in)	cse.19BCS4525@gmail.com
Student Contact No.	9123415629
Internship Organization Name	HighRadius
Organization Address	
Internship Supervisor	-
Internship Supervisor Phone	-
Internship Supervisor Email	-
Report period (start date)	28/01/2022
Report period (end date)	18/02/2022

1. Distribution of hours:

Orientation: 2 hours

Observing

Meetings (e.g. staffing, working with the team, etc)

Lectures, Seminars, Conferences ________

Assessment **1:20 Hours**

Planning (activity analysis, goals and objectives, etc) ________

Studying/Researching ________

C. Implementation (in *hours* which so ever is applicable. Otherwise mention Not Applicable):

a. Leadership NA

b. Counselling NA

c. Supervision 20 hours

d. Evaluation NA

e. Documentation 12 Hours

f. Discharge/Transition Plans NA

g. Other (Please specify) -

Total clock hours during this report period **48 Hours**

Introduction About the Company

I am working under High Radius as an intern. High Radius is a Fintech software company based on AI Autonomous Systems.

The HighRadius platform reduces cycle times in orders-to-cash process by automating receivables and payment processes across credit, e-billing and payment processing, deductions and collections.

I have been working with this company since 28/01/2022.

HighRadius offers cloud-based Autonomous Software for the Office of the CFO. More than 700 of the world's leading companies have transformed their order to cash, treasury and record to report processes with HighRadius. Our customers include 3M, Unilever, Anheuser-Busch InBev, Sanofi, Kellogg Company, Danone, Hershey's and many more.

Autonomous Software is data-driven software that continuously morphs its behavior to the ever-changing underlying domain transactional data. It brings modern digital transformation capabilities like Artificial Intelligence, Robotic Process Automation, Natural Language Processing and Connected Workspaces as out-of-the-box features for the finance & accounting domain.

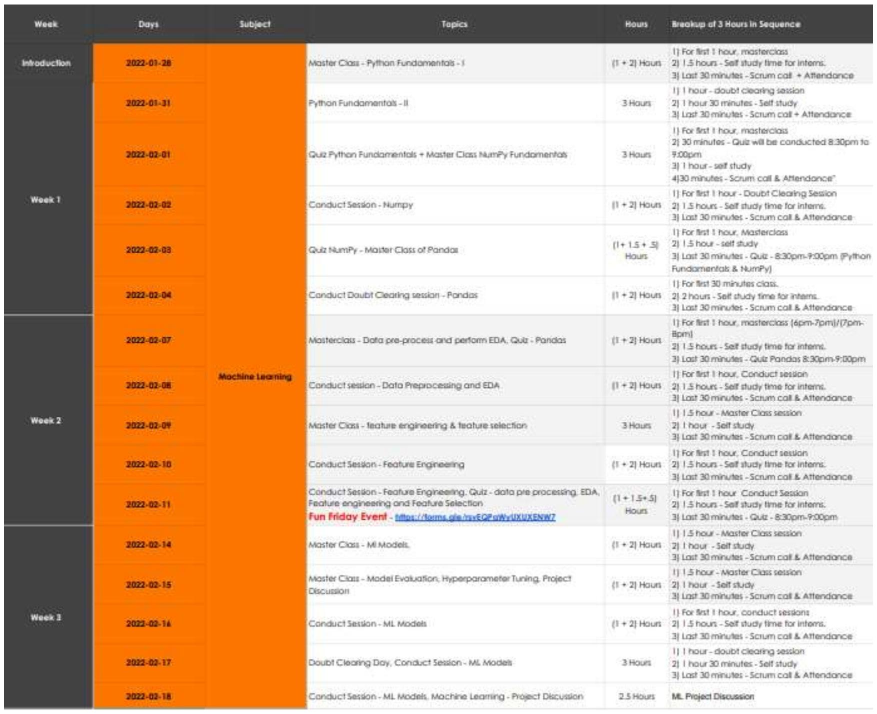
Finance business stakeholders have been led to believe that they have only two choices: pick an application software vendor that digitizes a paper or Excel-based process to an electronic system of record, or, choose a middleware platform for AI or RPA to build and maintain in-house, domain-specific capabilities. In contrast, HighRadius Autonomous Software combines the best of both worlds to deliver measurable business outcomes such as DSO reduction, working capital optimization, bad-debt reduction, reduce month close timelines and improve productivity in under six months.

Data-driven software that uses technologies like AI to continuously morph its behaviour based on the ever-changing underlying domain transactional data.

Accomplishments and Work Performed

1. During the initial stages of internship, we were given masterclasses in which we were taught the topics related to different the overall project to be made. The main topics which will be covered throughout this internship period was **Machine Learning**

1.Machine Learning



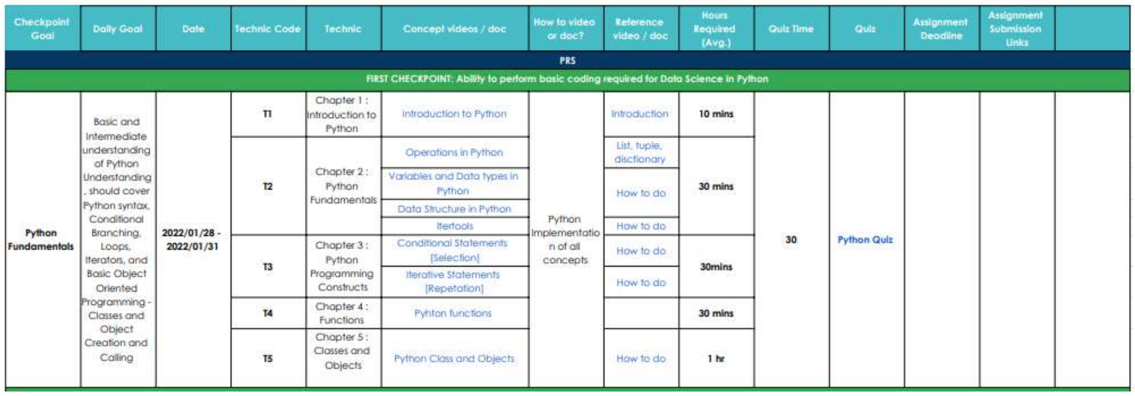
Week	Days	Subject	Topics	Hours	Breakup of 3 Hours in Sequence
Introduction	2022-01-28	Machine Learning	Master Class - Python Fundamentals - I	(1 + 2) Hours	1) For first 1 hour, masterclass 2) 1.5 hours - Self study time for interns. 3) Last 30 minutes - Scrum call + Attendance
Week 1	2022-01-31		Python Fundamentals - II	3 Hours	1) 1 hour - doubt clearing session 2) 1 hour 30 minutes - Self study 3) Last 30 minutes - Scrum call + Attendance
	2022-02-01		Quiz Python Fundamentals + Master Class NumPy Fundamentals	3 Hours	1) For first 1 hour, masterclass 2) 30 minutes - Quiz will be conducted 8:30pm to 9:00pm 3) 1 hour - self study 4)30 minutes - Scrum call & Attendance"
	2022-02-02		Conduct Session - Numpy	(1 + 2) Hours	1) For first 1 hour - Doubt Clearing Session 2) 1.5 hours - Self study time for interns. 3) Last 30 minutes - Scrum call & Attendance
	2022-02-03		Quiz NumPy - Master Class of Pandas	(1+ 1.5 + .5) Hours	1) For first 1 hour, Masterclass 2) 1.5 hour - self study 3) Last 30 minutes - Quiz - 8:30pm-9:00pm (Python Fundamentals & NumPy)
	2022-02-04		Conduct Doubt Clearing session - Pandas	(1 + 2) Hours	1) For first 30 minutes class. 2) 2 hours - Self study time for interns. 3) Last 30 minutes - Scrum call & Attendance
Week 2	2022-02-07		Masterclass - Data pre-process and perform EDA, Quiz - Pandas	(1 + 2) Hours	1) For first 1 hour, masterclass (6pm-7pm)/(7pm-8pm) 2) 1.5 hours - Self study time for interns. 3) Last 30 minutes - Quiz Pandas 8:30pm-9:00pm
	2022-02-08		Conduct session - Data Preprocessing and EDA	(1 + 2) Hours	1) For first 1 hour, Conduct session 2) 1.5 hours - Self study time for interns. 3) Last 30 minutes - Scrum call & Attendance
	2022-02-09		Master Class - feature engineering & feature selection	3 Hours	1) 1.5 hour - Master Class session 2) 1 hour - Self study 3) Last 30 minutes - Scrum call & Attendance
	2022-02-10		Conduct Session - Feature Engineering	(1 + 2) Hours	1) For first 1 hour, Conduct session 2) 1.5 hours - Self study time for interns. 3) Last 30 minutes - Scrum call & Attendance
	2022-02-11		Conduct Session - Feature Engineering, Quiz - data pre processing, EDA, Feature engineering and Feature Selection Fun Friday Event - https://forms.gle/rsvEQPaWyUXUXENW7	(1 + 1.5+.5) Hours	1) For first 1 hour Conduct Session 2) 1.5 hours - Self study time for interns. 3) Last 30 minutes - Quiz - 8:30pm-9:00pm
Week 3	2022-02-14		Master Class - Ml Models.	(1 + 2) Hours	1) 1.5 hour - Master Class session 2) 1 hour - Self study 3) Last 30 minutes - Scrum call & Attendance
	2022-02-15		Master Class - Model Evaluation, Hyperparameter Tuning, Project Discussion	(1 + 2) Hours	1) 1.5 hour - Master Class session 2) 1 hour - Self study 3) Last 30 minutes - Scrum call & Attendance
	2022-02-16		Conduct Session - ML Models	(1 + 2) Hours	1) For first 1 hour, conduct sessions 2) 1.5 hours - Self study time for interns. 3) Last 30 minutes - Scrum call & Attendance
	2022-02-17		Doubt Clearing Day, Conduct Session - ML Models	3 Hours	1) 1 hour - doubt clearing session 2) 1 hour 30 minutes - Self study 3) Last 30 minutes - Scrum call & Attendance
	2022-02-18		Conduct Session - ML Models, Machine Learning - Project Discussion	2.5 Hours	ML Project Discussion

As per the schedule shown in the above table, our internship was commenced from 28th January and for the machine learning part lasted till 18th February.

In this period, we started by learning basics of python in our masterclass which included theory and hands-on practice as well. These lectures were conducted on Zoom.

For the first checkpoint, we had started with Python basic.

1.1 First Checkpoint:

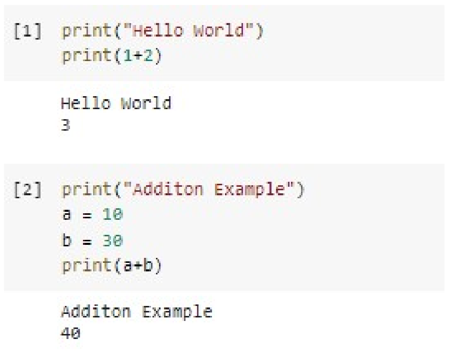


Checkpoint Goal	Daily Goal	Date	Technic Code	Technic	Concept videos / doc	How to video or doc?	Reference video / doc	Hours Required (Avg.)	Quiz Time	Quiz	Assignment Deadline	Assignment Submission Links	
PRS													
FIRST CHECKPOINT: Ability to perform basic coding required for Data Science in Python													
Python Fundamentals	Basic and Intermediate understanding of Python Understanding , should cover Python syntax, Conditional Branching, Loops, Iterators, and Basic Object Oriented Programming - Classes and Object Creation and Calling	**2022/01/28 - 2022/01/31**	**T1**	Chapter 1 : Introduction to Python	Introduction to Python	Python Implementation of all concepts	Introduction	**10 mins**	**30**	**Python Quiz**			
			T2	Chapter 2 : Python Fundamentals	Operations in Python		List, tuple, disctionary	**30 mins**					
					Variables and Data types in Python		How to do						
					Data Structure in Python								
					Itertools		How to do						
			T3	Chapter 3 : Python Programming Constructs	Conditional Statements [Selection]		How to do	**30mins**					
					Iterative Statements [Repetation]		How to do						
			T4	Chapter 4 : Functions	Pyhton functions			**30 mins**					
			T5	Chapter 5 : Classes and Objects	Python Class and Objects		How to do	**1 hr**					

Python is a Multi-Purpose programming language. It is used for developing GUI (Graphical User Interfaces), various scripting purposes, creating backend applications, web scraping and various other things. It is an Interpreted Language, that is, it is executed in a sequential manner and does not need to be compiled before it is executed. It is a strongly and dynamically typed programming language which is extendable and portable. It can be used to combine various programming languages together to work cohesively as one distinct entity. In addition to that, Python is also a free and open source programming language which means that it is free to use and everyone can contribute to its development.

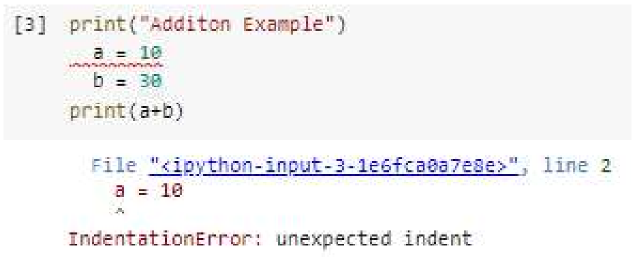
Python Fundamentals

Python is a very simple coding language that uses a very familiar language to code. It uses indentation to define blocks of code and they need to be consistent throughout the block.



```
[1]  print("Hello World")
     print(1+2)

     Hello World
     3

[2]  print("Additon Example")
     a = 10
     b = 30
     print(a+b)

     Additon Example
     40
```

The above example depicts the simplicity of python as a coding language. Indentation is very important in python and not following proper indentation structure causes an error.

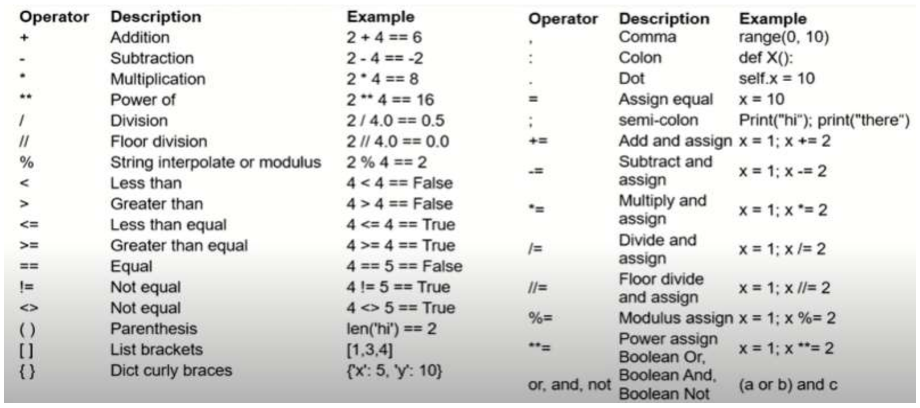


```
[3] print("Additon Example")
      a = 10
      b = 30
    print(a+b)

      File "<ipython-input-3-1e6fca0a7e8e>", line 2
        a = 10
        ^
    IndentationError: unexpected indent
```

Semicolons have almost no use in python but using them would not throw any error. It is not considered good practice while writing python code. It can be used to separate many commands in a single line.

Operators in Python

There are many operators in python that can be used for many purposes. They are stated below.



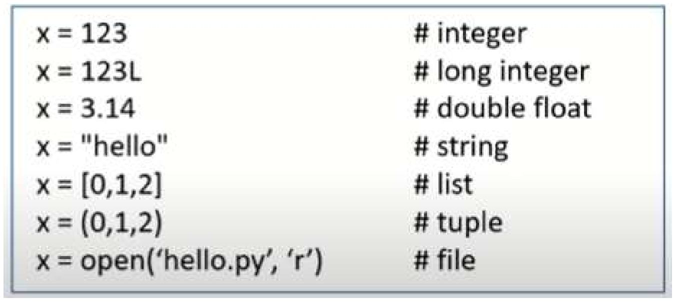
Operator	Description	Example
+	Addition	2 + 4 == 6
-	Subtraction	2 - 4 == -2
*	Multiplication	2 * 4 == 8
**	Power of	2 ** 4 == 16
/	Division	2 / 4.0 == 0.5
//	Floor division	2 // 4.0 == 0.0
%	String interpolate or modulus	2 % 4 == 2
<	Less than	4 < 4 == False
>	Greater than	4 > 4 == False
<=	Less than equal	4 <= 4 == True
>=	Greater than equal	4 >= 4 == True
==	Equal	4 == 5 == False
!=	Not equal	4 != 5 == True
<>	Not equal	4 <> 5 == True
( )	Parenthesis	len('hi') == 2
[ ]	List brackets	[1,3,4]
{ }	Dict curly braces	{'x': 5, 'y': 10}

Operator	Description	Example
,	Comma	range(0, 10)
:	Colon	def X():
.	Dot	self.x = 10
=	Assign equal	x = 10
;	semi-colon	Print("hi"); print("there")
+=	Add and assign	x = 1; x += 2
-=	Subtract and assign	x = 1; x -= 2
*=	Multiply and assign	x = 1; x *= 2
/=	Divide and assign	x = 1; x /= 2
//=	Floor divide and assign	x = 1; x //= 2
%=	Modulus assign	x = 1; x %= 2
**=	Power assign	x = 1; x **= 2
or, and, not	Boolean Or, Boolean And, Boolean Not	(a or b) and c

Variables and Data Types in Python

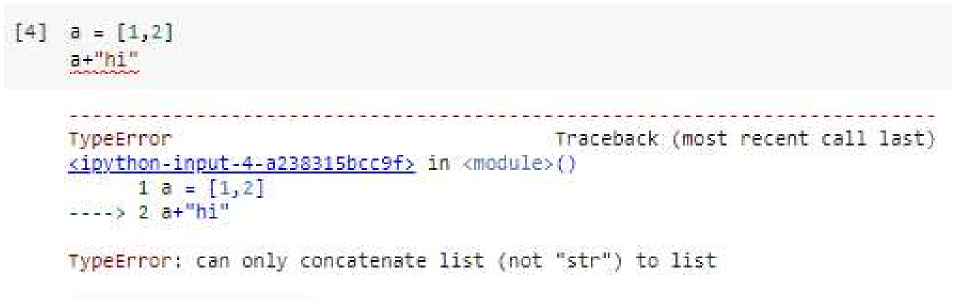
In Python, variables are considered as storage placeholders for texts and numbers.

Python is dynamically typed, such that there is no need to declare what the type of each variable is when it is declared or initialized [type() method is used to find the data type]



```
x = 123                  # integer
x = 123L                 # long integer
x = 3.14                 # double float
x = "hello"              # string
x = [0,1,2]              # list
x = (0,1,2)              # tuple
x = open('hello.py', 'r')   # file
```

Although you don't need to define the type of a variable, python is strongly typed in the sense that operations can not be performed between two dissimilar data types.



```
[4]  a = [1,2]
     a+"hi"

     ---------------------------------------------------------------------------
     TypeError                                 Traceback (most recent call last)
     <ipython-input-4-a238315bcc9f> in <module>()
           1 a = [1,2]
     ----> 2 a+"hi"

     TypeError: can only concatenate list (not "str") to list
```

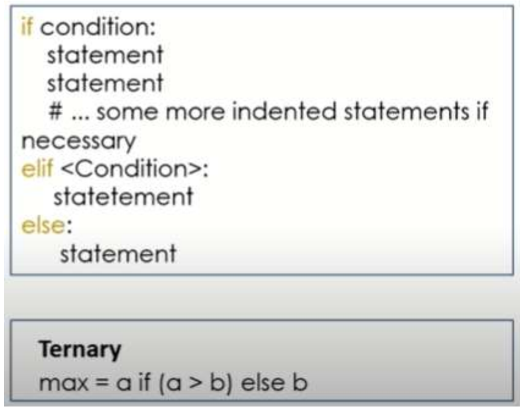
Python Programming Constructs

Constructs control the flow of the program. If we dive deep into the types of constructs, they are primarily of three types : Sequence, Selection and Repetition.

A **Sequence** is an order in which the code will get executed. **Selection** is the part where it is decided which block of code will get executed based on some conditions. **Repetition** is the construct that decides which part of the code will get executed multiple times based on specific criteria.

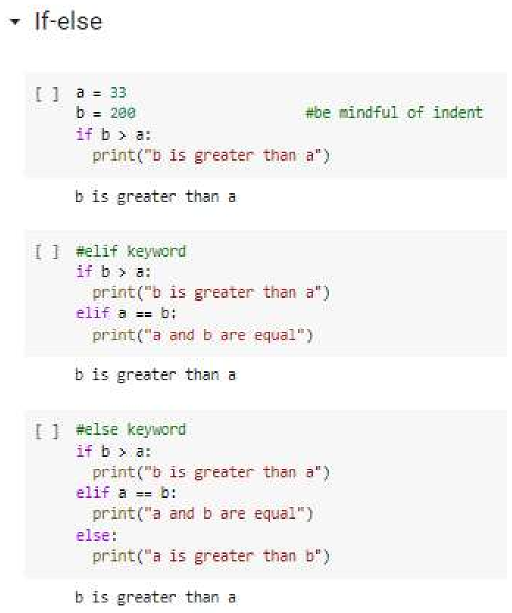
Conditional Statements [Selection]

Branching in Python can be achieved through the following keywords: if, elif (else-if) and else. The scope of the statement block is decided through indentation (cascading in case of nested conditions). An example of the construct can be seen in the following figure,



```
if condition:
   statement
   statement
   # ... some more indented statements if
necessary
elif <Condition>:
    statetement
else:
     statement

  Ternary
  max = a if (a > b) else b
```

An example of the construct in use can be found in the below code snippet,



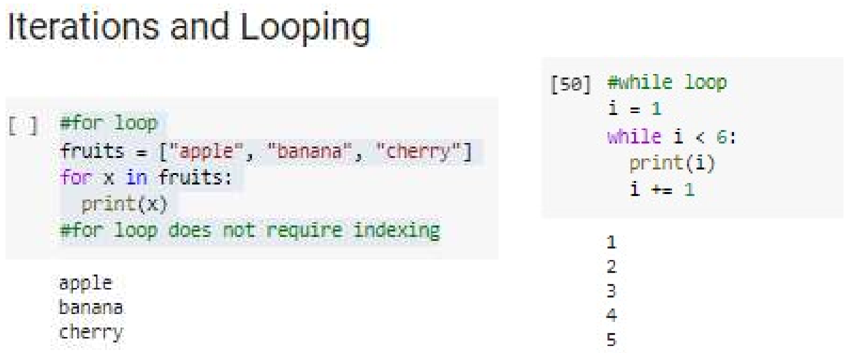
▾ If-else

```
[ ] a = 33
    b = 200                     #be mindful of indent
    if b > a:
      print("b is greater than a")

    b is greater than a

[ ] #elif keyword
    if b > a:
      print("b is greater than a")
    elif a == b:
      print("a and b are equal")

    b is greater than a

[ ] #else keyword
    if b > a:
      print("b is greater than a")
    elif a == b:
      print("a and b are equal")
    else:
      print("a is greater than b")

    b is greater than a
```

Here the score is compared and according to specific conditions (>90,>60 and <=90) different sets of code blocks are executed.

Iterative Statements [Repetition]

Iterative constructs in python are achieved through loops. They are primarily of two types: **for loop** and **while loop.**



Iterations and Looping

```
[ ] #for loop
    fruits = ["apple", "banana", "cherry"]
    for x in fruits:
      print(x)
    #for loop does not require indexing

    apple
    banana
    cherry
```

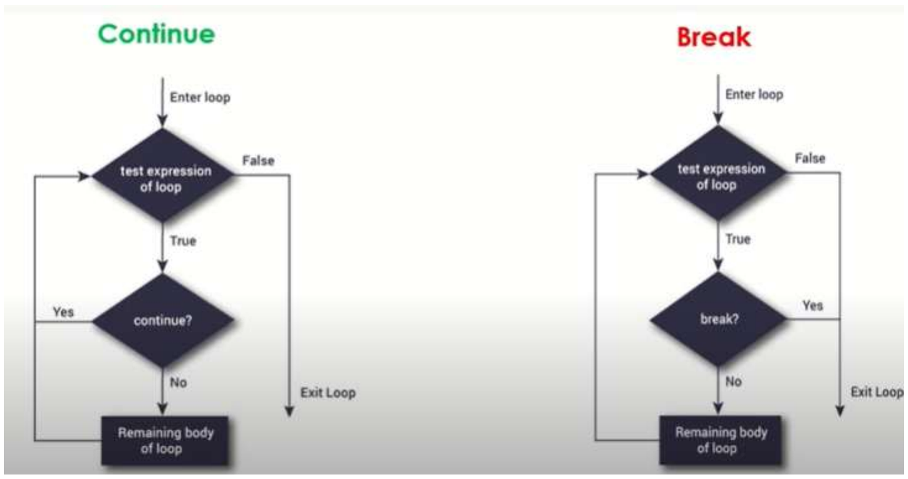
```
[50] #while loop
     i = 1
     while i < 6:
       print(i)
       i += 1

     1
     2
     3
     4
     5
```

The conditions in which the loops will continue to execute or stop after a specific number of iterations are controlled through two keywords, i.e., continue and break.

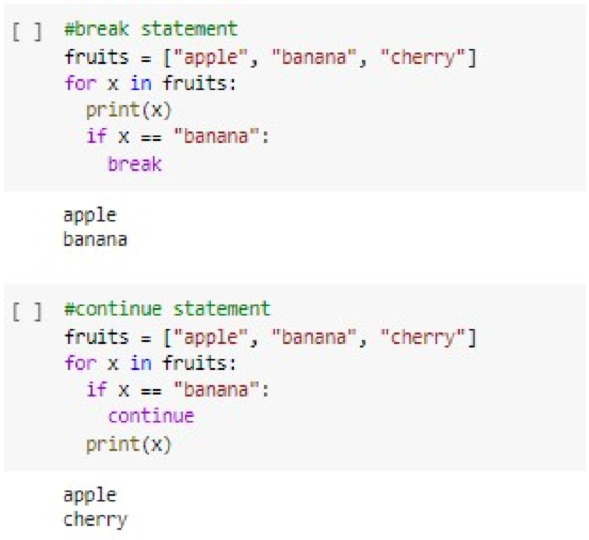
Continue statement is used to tell python to skip the rest of the statements in a current loop construct and continue with the next iteration of the code block.

Break, on the other hand, is used to completely break out of the loop.

The following figure shows the use of break and continue in separate programming constructs as they are used in python.



Use of break and continue in python:



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

    apple
    banana

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

    apple
    cherry
```

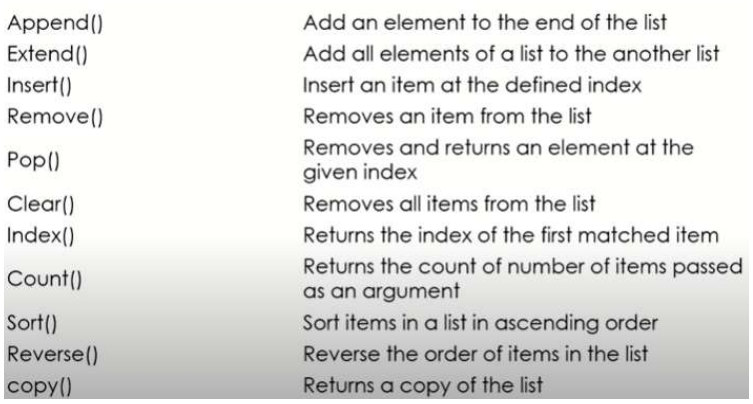
Data Structures in Python

There are many ways to store data in python. They are in the form of various data structures. For example, lists, tuples, dictionaries, sets, and many more.

1. **List**

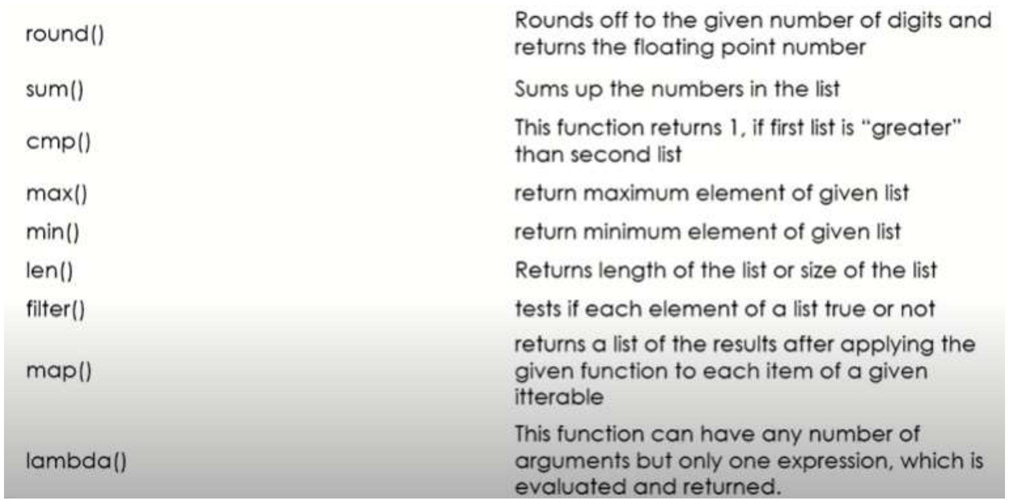
List is one of the simplest and most important data structures in python. They are defined by enclosing square brackets "[ ]" and each item is separated by a ",". Lists can be defined as a collection of items where each item has an assigned positional value (index value) starting from 0 (zero). It is mutable, i.e., its contents can be changed. It is similar to an array with some basic differences. For example, lists can store heterogeneous data types together under one name unlike matrices(arrays) that contain homogeneous data.

There are many **methods** that can be used to manipulate lists and do various operations. They are listed in the image below with their corresponding uses.



Append()	Add an element to the end of the list
Extend()	Add all elements of a list to the another list
Insert()	Insert an item at the defined index
Remove()	Removes an item from the list
Pop()	Removes and returns an element at the given index
Clear()	Removes all items from the list
Index()	Returns the index of the first matched item
Count()	Returns the count of number of items passed as an argument
Sort()	Sort items in a list in ascending order
Reverse()	Reverse the order of items in the list
copy()	Returns a copy of the list

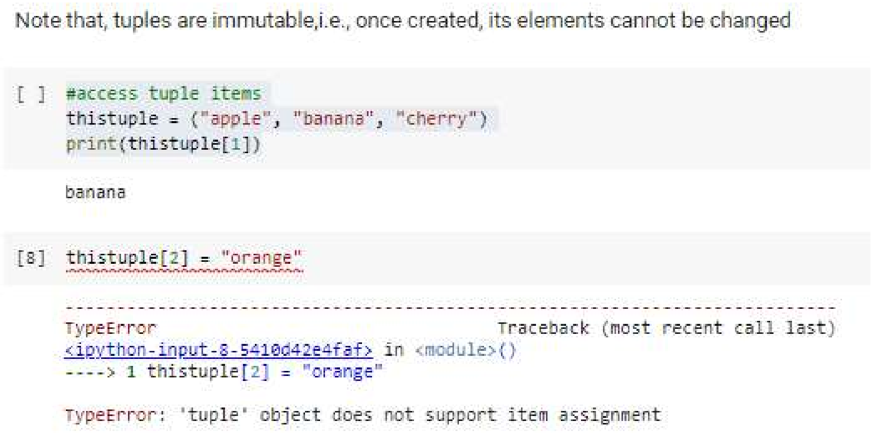
There are many **inbuilt functions** that are applicable for a list. They are as follows:



round()	Rounds off to the given number of digits and returns the floating point number
sum()	Sums up the numbers in the list
cmp()	This function returns 1, if first list is "greater" than second list
max()	return maximum element of given list
min()	return minimum element of given list
len()	Returns length of the list or size of the list
filter()	tests if each element of a list true or not
map()	returns a list of the results after applying the given function to each item of a given itterable
lambda()	This function can have any number of arguments but only one expression, which is evaluated and returned.

1. **Tuple**

A Tuple can be defined as an immutable list. It can not be altered. It is defined by initializing elements in between parentheses "( )". Once a tuple has been created, you can not add or alter elements in the tuple. It has only two methods: count() and index(). Count gives the frequency of a searched element while index provides the location of the searched element in the tuple (index starts with 0).

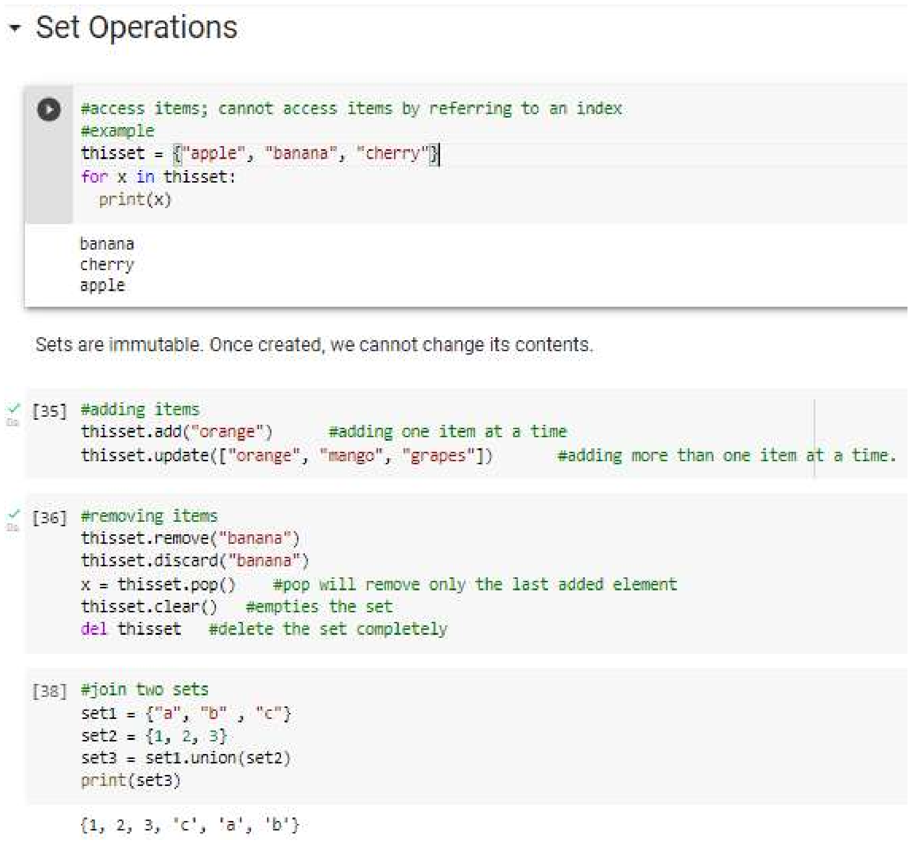


Note that, tuples are immutable,i.e., once created, its elements cannot be changed

```
[ ]  #access tuple items
     thistuple = ("apple", "banana", "cherry")
     print(thistuple[1])

     banana

[8]  thistuple[2] = "orange"

     ---------------------------------------------------------------------------
     TypeError                                 Traceback (most recent call last)
     <ipython-input-8-5410d42e4faf> in <module>()
     ----> 1 thistuple[2] = "orange"

     TypeError: 'tuple' object does not support item assignment
```

2. **Sets**

A set contains an unordered collection of unique and immutable objects. All kinds of operations that are applicable to a set can be used for sets.


Set Operations

```
#access items; cannot access items by referring to an index
#example
thisset = {"apple", "banana", "cherry"}
for x in thisset:
  print(x)
```

```
banana
cherry
apple
```

Sets are immutable. Once created, we cannot change its contents.

```
[35] #adding items
     thisset.add("orange")      #adding one item at a time
     thisset.update(["orange", "mango", "grapes"])      #adding more than one item at a time.
```

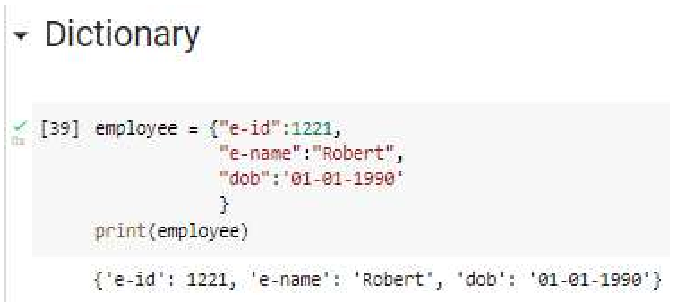
```
[36] #removing items
     thisset.remove("banana")
     thisset.discard("banana")
     x = thisset.pop()    #pop will remove only the last added element
     thisset.clear()   #empties the set
     del thisset   #delete the set completely
```

```
[38] #join two sets
     set1 = {"a", "b" , "c"}
     set2 = {1, 2, 3}
     set3 = set1.union(set2)
     print(set3)
```

```
{1, 2, 3, 'c', 'a', 'b'}
```

3. Dictionary

It is a python data structure that is used to store data in key-value pairs. They are a set of attributes that have corresponding values. It is an unordered, indexed, and changeable form of data that is written within curly braces.

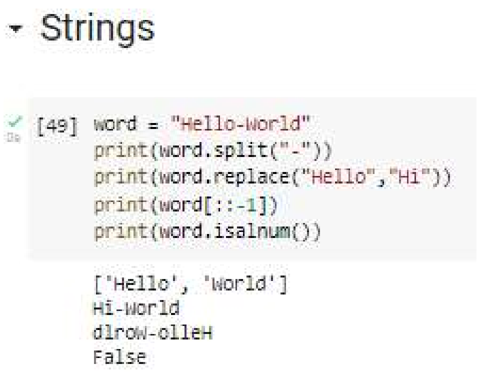

Dictionary

```
[39] employee = {"e-id":1221,
                 "e-name":"Robert",
                 "dob":'01-01-1990'
                 }
     print(employee)
```

```
{'e-id': 1221, 'e-name': 'Robert', 'dob': '01-01-1990'}
```

4. Strings

Strings can be defined as a list or an ordered chain of characters. We can perform various operations or manipulations on these strings.

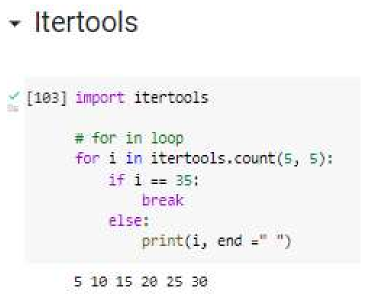


Strings

```
[49] word = "Hello-World"
     print(word.split("-"))
     print(word.replace("Hello","Hi"))
     print(word[::-1])
     print(word.isalnum())

['Hello', 'World']
Hi-World
dlroW-olleH
False
```

Itertools

Python's Itertool is a module that provides various functions that work on iterators to produce complex iterators. This module works as a fast, memory-efficient tool that is used either by itself or in combination to form complex algebraic equations.



Itertools

```
[103] import itertools

      # for in loop
      for i in itertools.count(5, 5):
          if i == 35:
              break
          else:
              print(i, end =" ")

5 10 15 20 25 30
```

Slicing Function

The Python slice() function allows us to slice a sequence. It means we can retrieve a part of a string, tuple, list, etc. We can specify the start, end, and step of the slice. The step lets you skip items in the sequence.

The **Syntax** of slice() is:

slice(start, stop, step)

slice() Parameters:

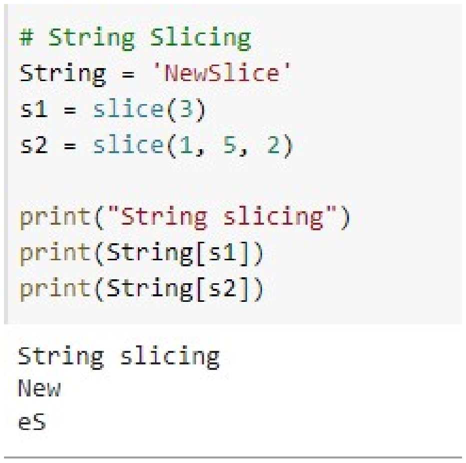
slice() can take three parameters:

- **start** (optional) - Starting integer where the slicing of the object starts. Default to None if not provided.
- **stop** - Integer until which the slicing takes place. The slicing stops at index stop -1 (last element).

- **step** (optional) - Integer value which determines the increment between each index for slicing. Defaults to None if not provided.

Return Type: Returns a sliced object containing elements in the given range only.

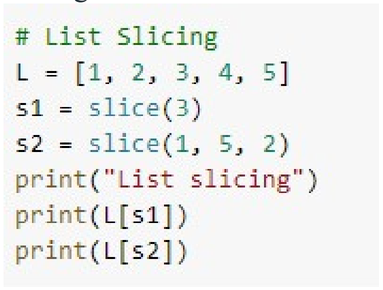
Slicing a string:



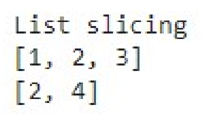
```
# String Slicing
String = 'NewSlice'
s1 = slice(3)
s2 = slice(1, 5, 2)

print("String slicing")
print(String[s1])
print(String[s2])
```

```
String slicing
New
eS
```

Slicing a List:

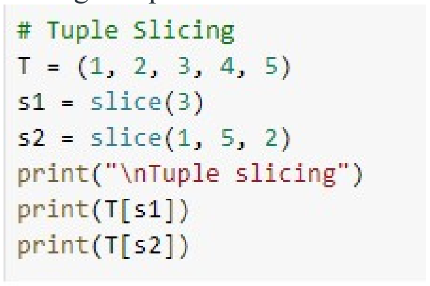


```
# List Slicing
L = [1, 2, 3, 4, 5]
s1 = slice(3)
s2 = slice(1, 5, 2)
print("List slicing")
print(L[s1])
print(L[s2])
```

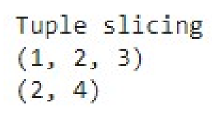


```
List slicing
[1, 2, 3]
[2, 4]
```

Slicing a tuple:



```
# Tuple Slicing
T = (1, 2, 3, 4, 5)
s1 = slice(3)
s2 = slice(1, 5, 2)
print("\nTuple slicing")
print(T[s1])
print(T[s2])
```



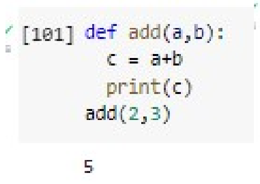
```
Tuple slicing
(1, 2, 3)
(2, 4)
```

Functions

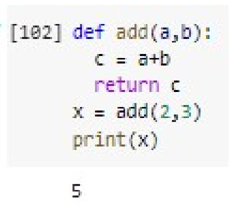
A function is a construct that is defined by the keyword “def”. The general syntax looks like this:

```
def  function_name(Parameter List):
        #Statements, i.e, the function body
        return statement (if required)
```

An example of a function used to add two numbers is given below,



```
[101] def add(a,b):
        c = a+b
        print(c)
      add(2,3)

5
```



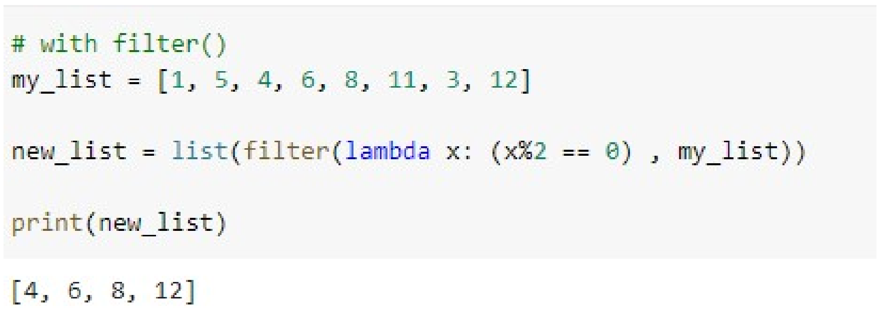
```
[102] def add(a,b):
        c = a+b
        return c
      x = add(2,3)
      print(x)

5
```

Lambda Function

We use lambda functions when we require a nameless function for a short period of time. In Python, we generally use it as an argument to a higher-order function (a function that takes in other functions as arguments). Lambda functions are used along with built-in functions like filter(), map() etc.

With filter():

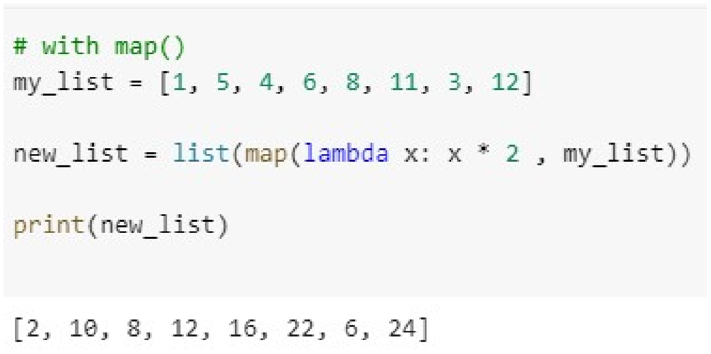
The filter() function in Python takes in a function and a list as arguments. The function is called with all the items in the list and a new list is returned which contains items for which the function evaluates to True.



```
# with filter()
my_list = [1, 5, 4, 6, 8, 11, 3, 12]

new_list = list(filter(lambda x: (x%2 == 0) , my_list))

print(new_list)

[4, 6, 8, 12]
```

With map():

The map() function in Python takes in a function and a list. The function is called with all the items in the list and a new list is returned which contains items returned by that function for each item.

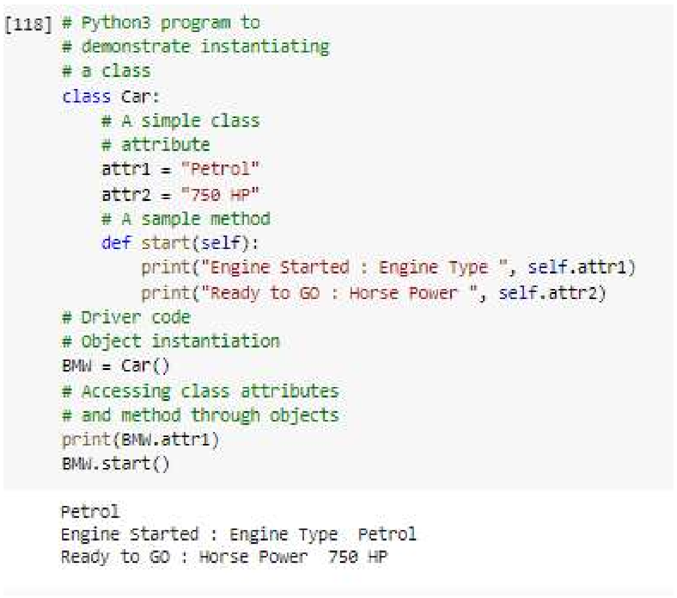


```
# with map()
my_list = [1, 5, 4, 6, 8, 11, 3, 12]

new_list = list(map(lambda x: x * 2 , my_list))

print(new_list)
```

```
[2, 10, 8, 12, 16, 22, 6, 24]
```

Classes and Objects

A class is a user-defined blueprint or prototype from which objects are created. Classes provide a means of bundling data and functionality together. Creating a new class creates a new type of object, allowing new objects of that type to be made. Each class instance can have attributes attached to it for maintaining its state. Class instances can also have methods (defined by their class) for modifying or manipulating their state.



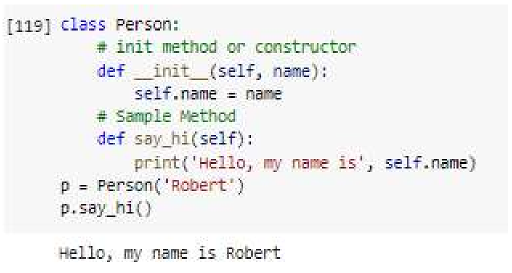
```
[118] # Python3 program to
      # demonstrate instantiating
      # a class
      class Car:
          # A simple class
          # attribute
          attr1 = "Petrol"
          attr2 = "750 HP"
          # A sample method
          def start(self):
              print("Engine Started : Engine Type ", self.attr1)
              print("Ready to GO : Horse Power ", self.attr2)
      # Driver code
      # Object instantiation
      BMW = Car()
      # Accessing class attributes
      # and method through objects
      print(BMW.attr1)
      BMW.start()
```

```
Petrol
Engine Started : Engine Type  Petrol
Ready to GO : Horse Power  750 HP
```

__init__ method

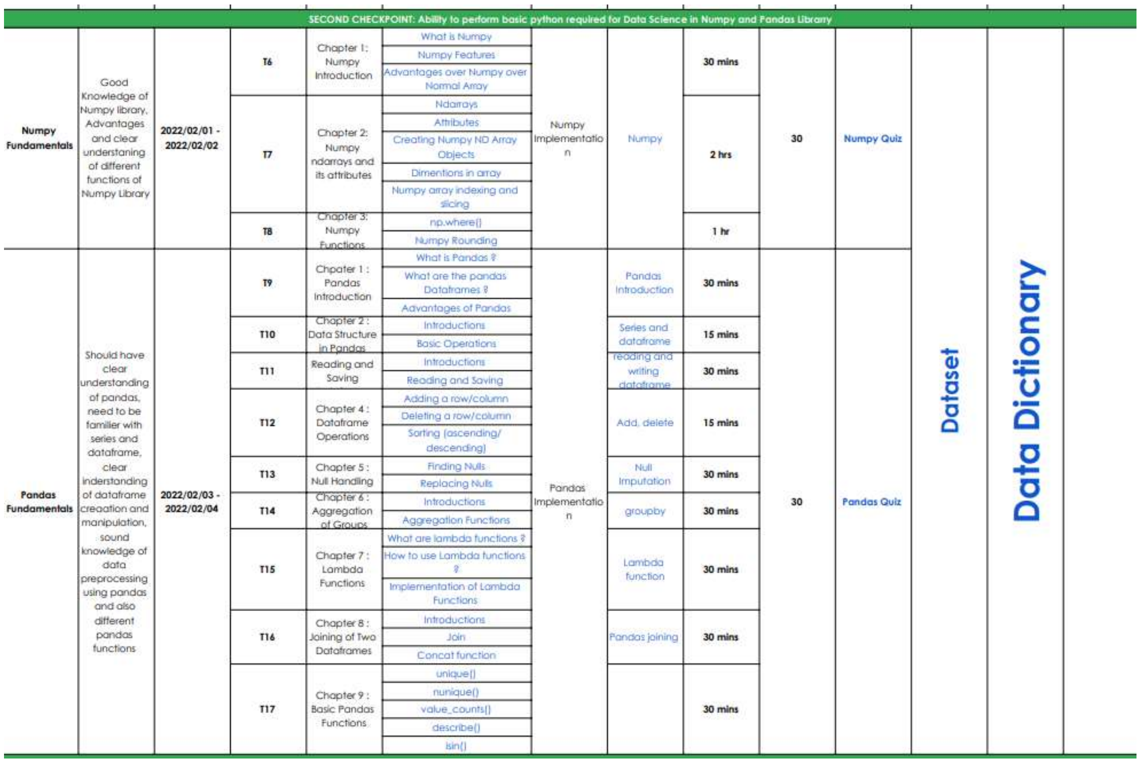
It is used to initialize the attributes for a class with specific values for a particular object. It is executed at the time of object creation for a particular class.

An example of the use of the __init__ function can be seen below,



```
[119] class Person:
          # init method or constructor
          def __init__(self, name):
              self.name = name
          # Sample Method
          def say_hi(self):
              print('Hello, my name is', self.name)
      p = Person('Robert')
      p.say_hi()

      Hello, my name is Robert
```

1.2 Second Checkpoint

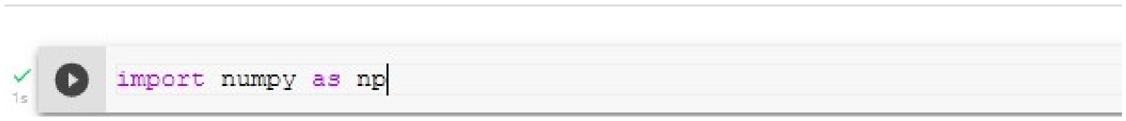


SECOND CHECKPOINT: Ability to perform basic python required for Data Science in Numpy and Pandas Library													
Numpy Fundamentals	Good Knowledge of Numpy library, Advantages and clear understaning of different functions of Numpy Library	**2022/02/01 - 2022/02/02**	**T6**	Chapter 1: Numpy Introduction	What is Numpy Numpy Features Advantages over Numpy over Normal Array	Numpy Implementation	Numpy	**30 mins**	**30**	**Numpy Quiz**	**Dataset**	**Data Dictionary**	
			T7	Chapter 2: Numpy ndarrays and its attributes	Ndarrays Attributes Creating Numpy ND Array Objects Dimentions in array Numpy array indexing and slicing			**2 hrs**					
			T8	Chapter 3: Numpy Functions	np.where() Numpy Rounding			**1 hr**					
Pandas Fundamentals	Should have clear understanding of pandas, need to be familier with series and dataframe, clear inderstanding of dataframe creaation and manipulation, sound knowledge of data preprocessing using pandas and also different pandas functions	**2022/02/03 - 2022/02/04**	**T9**	Chpater 1 : Pandas Introduction	What is Pandas ? What are the pandas Dataframes ? Advantages of Pandas	Pandas Implementation	Pandas Introduction	**30 mins**	**30**	**Pandas Quiz**			
			T10	Chapter 2 : Data Structure in Pandas	Introductions Basic Operations		Series and dataframe	**15 mins**					
			T11	Reading and Saving	Introductions Reading and Saving		reading and writing dataframe	**30 mins**					
			T12	Chapter 4 : Dataframe Operations	Adding a row/column Deleting a row/column Sorting (ascending/ descending)		Add, delete	**15 mins**					
			T13	Chapter 5 : Null Handling	Finding Nulls Replacing Nulls		Null Imputation	**30 mins**					
			T14	Chapter 6 : Aggregation of Groups	Introductions Aggregation Functions		groupby	**30 mins**					
			T15	Chapter 7 : Lambda Functions	What are lambda functions ? How to use Lambda functions ? Implementation of Lambda Functions		Lambda function	**30 mins**					
			T16	Chapter 8 : Joining of Two Dataframes	Introductions Join Concat function		Pandas joining	**30 mins**					
			T17	Chapter 9 : Basic Pandas Functions	unique() nunique() value_counts() describe() isin()			**30 mins**					

Numpy is a library for the python programming language adding support to large, multi-dimensional arrays and matrices along with a large collection of high-level mathematical functions to operate on these arrays.

Numpy can deal with N-dimensional arrays.

To use Numpy in Python, we can import the numpy package as follows:



```
import numpy as np
```

Why use Numpy?

In Python we have lists that serve the purpose of arrays, but they are slow to process.
Numpy aims to provide an array object that is up to 50x faster than traditional Python lists.
The array object in Numpy is called **ndarray**, it provides a lot of supporting functions that make working with ndarray very easy.
Arrays are very frequently used in data science, where speed and resources are very important.

Numpy Features

Numpy is useful when it comes to array manipulation. Following table lists some features of numpy used for array creation and manipulation.

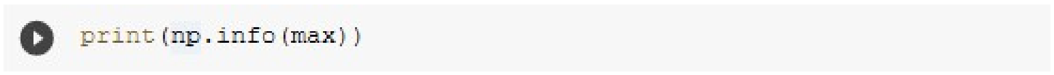
Feature	Description
Numpy 1-D Array	Making 1D array
Numpy 2-D Array	Making 2D array
Array Multiplication	Multiplying 2 or more array
numpy.ones	Matrix filled with ones
numpy.zeros	Matrix filled with zeros
numpy.random	Matrix filled with random numbers
numpy.arange	Create array with increments of a fixed step size
numpy.linspace	Create array of fixed length
numpy.full	Create a constant array of any number ‘n’
numpy.tile	Create a new array by repeating an existing array for a particular number of times
numpy.eye	Create an identity matrix of any dimension

numpy.random.randint	Random integer
Numpy 3-D Array	Making 3-D array

Advantages of Numpy over Normal Array

- Numpy uses much less memory to store data
- It allows creation of N-dimensional arrays
- Mathematical operations on Numpy n-dimensional arrays
- More powerful slicing and Broadcasting functionality
- Efficient Data Representation

Numpy provides a help function, providing the documentation for its methods, functions, classes and modules, by using the .info() function**.**



```
print(np.info(max))
```

What is Pandas?

Pandas is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the_python programming language.

Pandas is quite a game changer when it comes to **analyzing data** with Python and it is one of the most preferred and widely used tools in_**data munging/wrangling**. Pandas is an open source, free to use and it was originally written by Wes McKinney .

What's cool about Pandas is that it takes data (like a CSV or TSV file, or a SQL database) and creates a Python object with rows and columns called data frame that looks very similar to a table in a statistical software (like Excel).

Importing Pandas :

After the pandas have been installed into the system, you need to import the library. This module is generally imported as:

```
import pandas as pd
```

Here, pd is referred to as an alias to the Pandas. However, it is not necessary to import the library using the alias, it just helps in writing less code every time a method or property is called.

What are Pandas Data Frames?

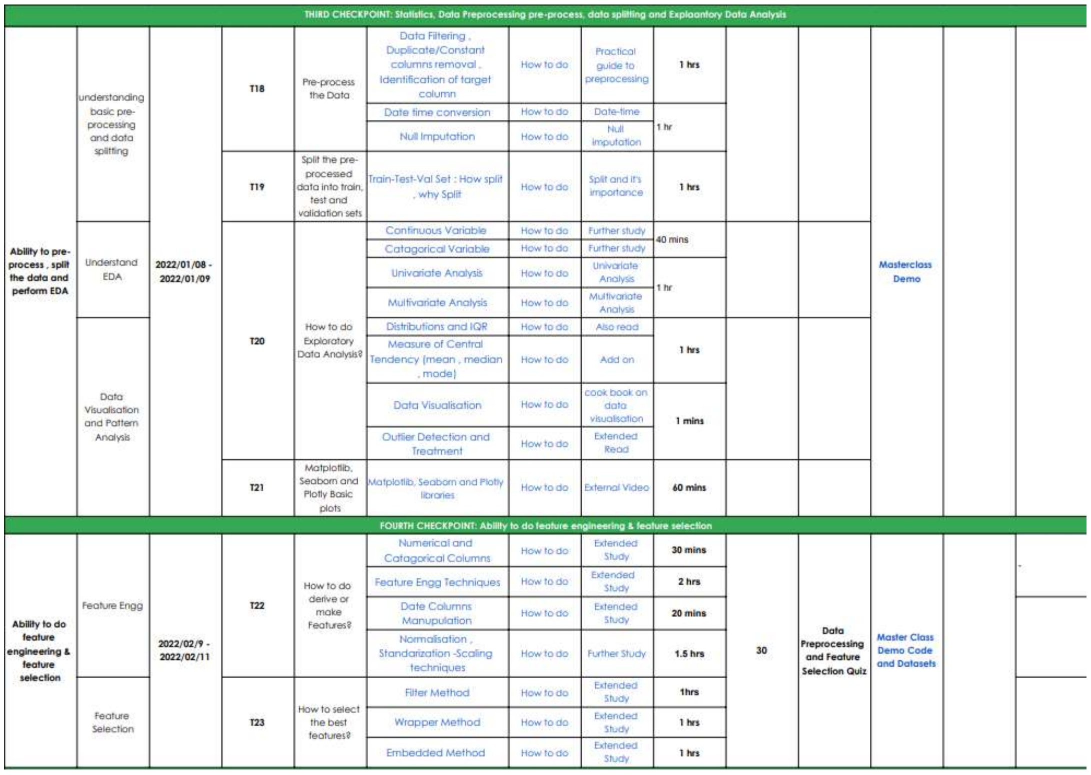
Pandas DataFrame is a two-dimensional size-mutable, potentially heterogeneous tabular data structure with labeled axes (rows and columns).

In general, we can say that the Pandas DataFrame consists of three main components: the data, the index, and the columns. DataFrames are extremely important going forward, as we can read & store excel sheets into DataFrames and use many manipulation techniques on them, as we'll learn ahead.

Advantages of Pandas

1. Fast and efficient for manipulating and analyzing data.
2. Data from different file objects can be loaded.
3. Easy handling of missing data (represented as NaN) in floating point as well as non-floating point data
4. Size mutability: columns can be inserted and deleted from DataFrame and higher dimensional objects
5. Data set merging and joining.
6. Flexible reshaping and pivoting of data sets
7. Provides time-series functionality.
8. Powerful group by functionality for performing split-apply-combine operations on data sets.

1.3 Third and Fourth Checkpoint



THIRD CHECKPOINT: Statistics, Data Preprocessing pre-process, data splitting and Explaantory Data Analysis													
Ability to pre-process , split the data and perform EDA	understanding basic pre-processing and data splitting	2022/01/08 - 2022/01/09	T18	Pre-process the Data	Data Filtering , Duplicate/Constant columns removal , Identification of target column	How to do	Practical guide to preprocessing	1 hrs			Masterclass Demo		
					Date time conversion	How to do	Date-time	1 hr					
					Null Imputation	How to do	Null imputation						
			T19	Split the pre-processed data into train, test and validation sets	Train-Test-Val Set : How split , why Split	How to do	Split and it's importance	1 hrs					
	Understand EDA		T20	How to do Exploratory Data Analysis?	Continuous Variable	How to do	Further study	40 mins					
					Catagorical Variable	How to do	Further study						
					Univariate Analysis	How to do	Univariate Analysis	1 hr					
					Multivariate Analysis	How to do	Multivariate Analysis						
	Data Visualisation and Pattern Analysis				Distributions and IQR	How to do	Also read	1 hrs					
					Measure of Central Tendency (mean , median , mode)	How to do	Add on						
					Data Visualisation	How to do	cook book on data visualisation	1 mins					
					Outlier Detection and Treatment	How to do	Extended Read						
			T21	Matplotlib, Seaborn and Plotly Basic plots	Matplotlib, Seaborn and Plotly libraries	How to do	External Video	60 mins					
FOURTH CHECKPOINT: Ability to do feature engineering & feature selection													
Ability to do feature engineering & feature selection	Feature Engg	2022/02/9 - 2022/02/11	T22	How to do derive or make Features?	Numerical and Catagorical Columns	How to do	Extended Study	30 mins	30	Data Preprocessing and Feature Selection Quiz	Master Class Demo Code and Datasets		-
					Feature Engg Techniques	How to do	Extended Study	2 hrs					
					Date Columns Manupulation	How to do	Extended Study	20 mins					
					Normalisation , Standarization -Scaling techniques	How to do	Further Study	1.5 hrs					
	Feature Selection		T23	How to select the best features?	Filter Method	How to do	Extended Study	1hrs					
					Wrapper Method	How to do	Extended Study	1 hrs					
					Embedded Method	How to do	Extended Study	1 hrs					

What is a Target Variable?

The target variable of a dataset is the feature of a dataset about which you want to gain a deeper understanding. A supervised machine learning algorithm uses historical data to learn patterns and uncover relationships between other features of your dataset and the target.
The target variable will vary depending on the business goal and available data. For example, let's say you want to use sentiment analysis to classify whether tweets about your company's brand are positive or negative. Some aspects of a tweet that can be useful as features are word tokens, parts of speech, and emoticons. A model cannot learn how those features relate to sentiment without first being given examples of which tweets are positive or negative (the target).

Importance of Target Variables:

Without a labelled target, supervised machine learning algorithms would be unable to map available data to outcomes, just as a child would be incapable of figuring out that cats are called "cats" without having been told so at least a few times. It is important to have a well-defined target since the only thing an algorithm does is learn a function that maps relationships between input data and the target. The model’s outcomes will be meaningless if your target doesn’t make sense.

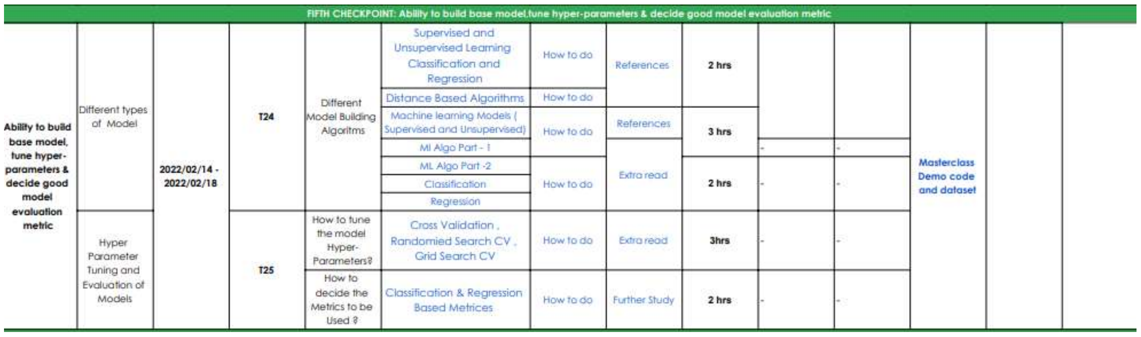
Feature engineering is the process of using domain knowledge of the data to create **features** that make machine learning algorithms work.
What does the term feature mean here?
Ex- How do you make decision to buy a car? You go through some of the attributes of different cars and then you make a decision to buy or not based on your understanding. These features can be min and max speed, seating capacity etc.
What is feature engineering then?
If you are given total area of car, torque produced, in depth details of engine etc, you may not understand it well. That’s why manufacturers present a more understandable entities like min-max range of speed, seating capacity, mileage etc.
This is nothing but deriving/ creating attributes which can be understood easily and can help people to make decision.
Similarly machine learning algorithms work better if we can feed attributes which cause a particular outcome to be predicted by model.

A tabular dataset contains multiple fields/attributes. These attributes are called the raw features. Majorly the data type for these fields are numerical, categorical and date time.

Different kind of feature engineering techniques are applied for different kind of data types.

Objective:
This play will help you to do feature engineering on numerical columns such as amount, number of days, age, weight etc.

1.4 Fifth Checkpoint



FIFTH CHECKPOINT: Ability to build base model,tune hyper-parameters & decide good model evaluation metric													
Ability to build base model, tune hyper-parameters & decide good model evaluation metric	Different types of Model	**2022/02/14 - 2022/02/18**	**T24**	Different Model Building Algoritms	Supervised and Unsupervised Learning Classification and Regression	How to do	References	**2 hrs**			**Masterclass Demo code and dataset**		
					Distance Based Algorithms	How to do							
					Machine learning Models (Supervised and Unsupervised)	How to do	References	**3 hrs**					
					Ml Algo Part - 1				-	-			
					ML Algo Part -2	How to do	Extra read	**2 hrs**	-	-			
					Classification								
					Regression								
	Hyper Parameter Tuning and Evaluation of Models		**T25**	How to tune the model Hyper-Parameters?	Cross Validation , Randomied Search CV , Grid Search CV	How to do	Extra read	**3hrs**	-	-			
				How to decide the Metrics to be Used ?	Classification & Regression Based Metrices	How to do	Further Study	**2 hrs**	-	-			