Python (Programming language):

Introduction:

Python are the popular, easy-to-learn programming language use for web development, data science, AI, Automation and more. Known for its simple syntax and vast libraries.

Topic Cover:

1. **Basic concept:**
    a. Variable
    b. Data type
    c. Conditional statement
    d. Loop
    e. Data structure
    f. Function
    g. Object originated programming
2. **Libraries for AI:**
    a. For File handling
        (a) Numpy
        (b) Pandas
    b. For Visualization
        (a) Matplotib
        (b) Seaborn
    c. For ML
        (a) Scikit-learn
    d. For deep learning
        (a) TensoFlow / PyTorch

Why Python is best:

1. Easy and simple

2. Free and open source
3. High level language
4. Portable (mean your code can run on multiple platform like window, macOS, and Linux without needing changes).

Lecture #01:

Topic:

1. Variable
2. Data type
3. Input and output
4. Operator
5. Type conversation and type casting
6. And other

First Program:

```
print("hello word")
```

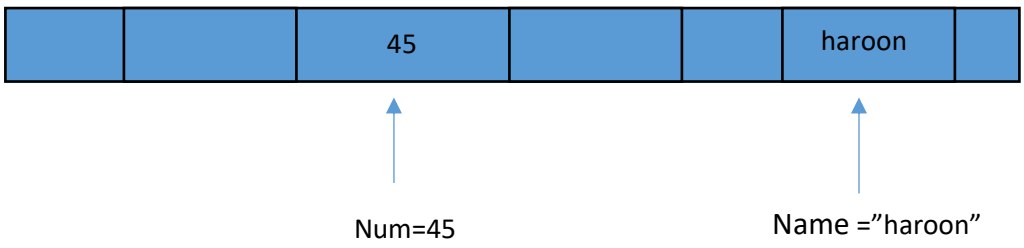
Output:

hello word

Variable:

Variable is simply a name of place in memory that contain some value and that value is changeable

Memory view:



Syntax:

Num=45

Percentage=45.5

Name=”haroon”

Notes: variable is only a name of container that contains some changeable value. Also called identifier

Node: Variable string are also create with single and triple course like ‘haroon’ and “””haroon”””)

Rule for identifier or variable:

1. For name of variable use only:
   A. A-Z
   B. a-z
   C. 0-9
   D. underscores (_)
2. Dose not start with number (0-9)
3. Does not use specific symbol (-, +, = etc)
4. Can be any length
5. Keyword are not use for the name of variable like while, for break etc.

Output in Python:

The print() function is use to print the string or any value

Syntax

print(“welcome to python”)

print(“the vale is “, variable_name)

Example #01:

```
print("That are the my first's lecture", 'I Am form "university of AJK" ')
num =34
print("the val is ", num)
```

Output

That are the my first's lecture I Am form "university of AJK"

the val is 34

Type() Function:

This function is use to print or get the data type of the variable.

syntax

print(type(variable_name))

Data type:

There are many built-in data type that are normally use to stored data.

Common data type:

1. Integers (+ve, -ve, and zero)
2. String (string or word)
3. Float (stored the floating value)
4. Boolean (Two possible value true and false)

Example #02

```
roll_Num=35
marks=56.6
name = "haroon"

print(roll_Num,type(roll_Num))
print(marks, type(marks))
print(name, type(name))
```

Sum of two number:

```
# sum of two number
num1=34
num2=45
sum=num1+num2
print("sum of ","num 1 =", num1,"num 2 =",num2," is " ,sum)
```

Comment:

Comment in python is a single or multiple line that does not execute (programmer are simple write for his understanding)

Single line comment:

#This is single line comment

Multi line comment:

“””

This is multi line comment

For understanding

“””

Input

The input() function is use to take input form the user

Syntax:

Num=input (“Enter the number”)

Operator:

They are simple symbol or characters that are used to perform a specific operation like sum, subtraction, multiplication and division

Type of operator:

1. Arithmetic operation (+, -, *, /, %, **)
2. Relational operator (=, !=, >, <, <=, <=, True, False)
3. Assignment operator (=, +=, -+, *=, /=, %=, **=)
4. Logical operation (not, and, or)

Type Conversion and Type Casting

As when we get input in python the input() function give string input to convert it into integer or floating we convert the string into the required data type.

Type Conversion:

Changing are automatically by compiler when they occur in expression

Like

print(2+2.02) #4.04

Type Casting:

The data type of some variable may cast or change to the other data type using there function like for int we use int() function.

Example #01(a):

Int() -> Other data type to integer

Float() -> Other data type to float

Str() -> Other to string

Example #03:

WAP to take two number as input and print following:

Raised second number to power of first number, increment first number by two and find the reminder of second number with first.

```
num=None
num=int(input("Enter the 1st number  "))

num1=None
num1=int(input("Enter the 1st number  "))

print ("second raised to power", num**num1)
print("increment first number by two ", num+2)
print("the reminder of second number with first is =", num1%num)
```

Eval Function:

Automatically convert string into input data type:

Eval function is use for this purpose.

Syntax:

eval(input(“Enter the number”))

Example #01.4

```
#Automatically find the data type of input number
num=eval(input("Enter number "))
print(num)
print(type(num))
```

Lecture #01 code

Lecture #02:

Topic:

1. String
2. Conditional statement

Node: String are also create with single and triple course like ‘haroon’ and “””haroon”””) and single letter are also support like ‘A’

String:

String are the built-in data type that are used to store sequence of characters.

Syntax:

Str_name=”value”

Example #02.1:

```
Name="haroon"
print(name)
```

Escape Sequence Character:

Character	Use
\n	They are used for shift the control in next line
\t	They are used for add a tab space

Basic Operation for String:

1. **Concatenation**

We can simple add two string but does not subtract and divide or multiply

String_name=str1+str2

2. **Length of string**

The Len () function is used to print or get the length of the string

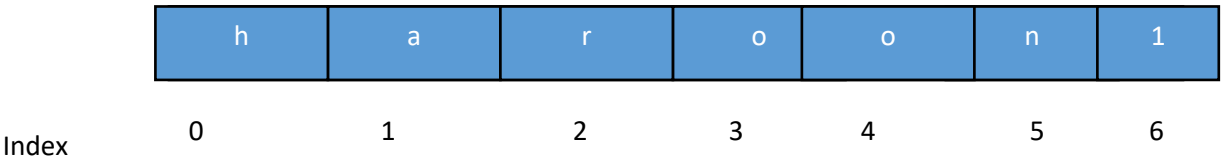
len(string_name)

Example #02.2:

```
#Example #02.2 Basic operation
first_name="muhammad"
last_name="Haroon"
full_name=first_name+" "+last_name
print(full_name)
print(len(full_name))
```

Index:

Index are the position number of single character in string



String_name=”haroon1”

Character of String:

String are immutable mean we cannot change individual character in string by using index

Example #02.3

```
#Example #02.3 index and immutable  characters
name="haroon1"
print(name[3])  #o

name+="rana"
print(name)  #haroon1rana

name[2]='h'
print(name[2])    #generate an error
```

String Slicing:

Slicing mean cut or split the string and access a specific part of string using index.

Syntax:

String_name [start_idx : end_idx]

Negative index slicing:

We can also split the string or access a specific part of string using negative index

h	a	r	o	o	n	1
-7	-6	-5	-4	-3	-2	-1

Index

Syntax:

String_name [start_idx : end_idx]

Only negative index are used syntax are same but sequence of character of word are different

End_idx value are not print or ignore in the resulting of the string splicing

Last value are not accessible using negative index slicing

Example #02.4

```
#Example #02.4 String Slicing
Name="haroon1"
print(Name[0:6])    #haroon

onlyName=Name[0:6]
print(onlyName)     #haroon

print(Name[-7:-1])   #haroon
```

String Function:

1. **String_Name.endwith("str")**

   Return the true and false value accordant to the occurrence of that letter

2. **String_name.capitalize()**

   Give a new string with the capitalize first characters only, but the original string remain in change.

3. **String_name.replace(old_string, new_string)**

   Return new string by changing the old_string into new_string.

4. **String_name.find("particular_word_Or_Characters")**

   To find a particular word or character in a string. They give the starting index of the finding word or character.

5. **String_name.count("word_or_character")**

   Count the occurrence of specific word or letter in a string.

Example #02.5

```
string_name="we are study python form univeristy of azad jammu and kashmir,
Pakistan"
print(string_name.endswith("tan"))      #Ture

string_intro=string_name.replace(string_name, "I am a student of CS")   #return
new string
```

```
print(string_intro)
print(string_name.find("python"))      #starting index of the find word

print(string_name.count('j'))       #count the occurrence of a particular word or
character.
```

Conditional Statement:

The condition statement are used to change the sequence of the execution based on a particular condition.

Syntax:

If (condition)

#Block of Code

Else

#Block of Code

Proper spacing or indentation are important in python because when a particular condition is true or false the statement are executer that a start with the tab space from the if statement. If proper spacing is not use than they will not perform a required task.

Example #02.6

```
# Example #02.6 conditional statement

age =int(input("Enter your age "))
if(age>=18):
    print("Apply for ID cart", "and Can vote")
else:
    print("Does not eligible for ID cart")
    print("Can not vote")

print("The End")
```

Multiple if_Statement:

We can also use multiple if statement other if statement is define by elif keyword.

Syntax:

If (condition)

#Block of Code

Elif (condition)

#Block of Code

Else

#Block of Code

Only one if and else are allow but we can used multiple elif statement to check another condition.

Example #02.7

```
# Example #02.7 multiple if-else statement
marks =int(input("Enter your marks "))
if(marks>100):
    print("Enter the valid marks ")
elif(marks>=90 and marks<=100):
    print("Supper")
elif(marks>=80 and marks<90):
    print("veryGood")
elif(marks>=70 and marks<80):
    print("Good")
elif(marks>=50 and marks<70):
    print("Fair")
else:
    print("Fail")
```

Nested if_else:

We can also write if statement with in another if statement this concept is known as nested if_else.

Syntax:

If (condition)

If(condition)

#Block of Code

Else

#Block of Code

Else

#Block of Code

Example #02.8

```
#Example #02.6 check give number is positive or negative than check is they even
or odd
num=int(input("Enter a number "))
if(num>=0):
    if(num%2==0):
        print("Number is positive and even")
    else:
        print("Number is positive and odd")
else:
    if(num%2==0):
        print("The number is negative and even")
    else:
        print("The number is negative and odd")
```

[Lecture #02 Code]

Lecture #03:

Topic:

1. List

2. Tuple

List in Python:

List are built-in data type that are used to store the collection of element of any data type.

Syntax:

List_name= [“value”, 1 “value”, 2]

Example #03.1:

```
student=[35, "Muhammad Haroon", 450]
print(student)   #[35, 'Muhammad Haroon', 450]
print(student[1])    #Muhammad Haroon
print(type(student))
```

List slicing:

Slicing mean cut or split the list and make a list of specific part of list using index.

Syntax:

List_name [starting_idx, ending_idx]

The ending index value are not included

And negative index slicing are also work in string slicing

Example #03.2

```
# Example 02 Slicing
roll_number=[23,45,65,32,12,67,87]
roll_number.sort()
Roll_num=roll_number[3:6]
print(Roll_num)

# negative index Slicing
```

```
roll_num_negative=roll_number[-5:-1]
print(roll_num_negative)
```

List Operation:

List are mutable mean we can change the individual index value

All the method and operation change the original list. And they return a none value.

1. **list_name.append("value")**

   Add new value at the end of list

2. **list_name.sort()**

   Sort the element of the list in ascending.

3. **list_name.sort(reverse=true)**

   Sort element in descending order

4. **list_name.reverse()**

   Reverse the list

5. **list_name.insert(idx, "value")**

   Insert new element at a specific index.

6. **List_name. Remove("value")**

   The will delete or remove the value that are specified in the argument.

7. **List_name. Pop(idx)**

   Delete value from a specific index as mention in the function called argument

Example #03.4

```
#Example 03 list function
student=[35,'haroon rana', 500]
student.append("University of Ajk")   #Add new element in list
student.insert(1,"Student of CS")
print(student.pop(3))
print(student)
```

Tuple in python:

Tuple are the immutable built-in data type used to store a sequence of data of different data type.

Syntax:

tuple_name= (“value”, 1 “value”, 2)

Example #03.4

```
student=("haroon", 35, 421)
print(student[1])
```

Tuple slicing:

Slicing mean cut or split the tuple and make a tuple of specific part of tuple using index.

Syntax:

tuple_name (starting_idx, ending_idx)

The ending index value are not included

And negative index slicing are also work in tuple slicing

In tuple () this bracket are used to initialize the tuple and [] are used for list

Example #03.5

```
# Example 05 Slicing in tuple
student=("haroon", 35, 421)
print(student[:2])
print(student[:-1])
```

Tuple Operation:

1. **tuple_name. index(“value”)**

   Give the index of specific value in tuple

2. **tuple_name. count(“value”)**

   Count the occurrence of specific value in the tuple.

Example #06.6

```
# Example 06 tuple function
tup=(2,4,5,3,6,7,8,66,3,9)
print(tup.index(66))
print(tup.count(3))
```

[Lecture #03 code]

Lecture #04

Topic:

1. Dictionary
2. Set

Dictionary:

Dictionary in python used to stored data by specific **key: value** pairs

Syntax:

Dictionary_name= {

"key" : "value" ,

"key2" : "value"

}

The value in dictionary are any data type integer float, sting list or tuple

Key name are also a numerical number

Example #04.1:

```
# Example #01 Dictionary
student={
    "name":"muhammd haroon",
    "Roll_num":35,
    "marks":431
}
print(student)
```

Characters of Dictionary:

1. **Un order**

   The dictionary element are print or access in random order mean sequence of element or key value pair change after each execution.

2. **Mutable**

   Mean the value of individual key can be change able

3. **Dose not allow duplicate key for two value**

Element Access in Dictionary:

The individual value is access by using its key

Syntax:

Dictionary_name [“key”]

Example #04.2

```
# Example #02 value accessing in Dictionary
student={
    "name":"muhammd haroon",
    "Roll_num":35,
    "marks":431
}
print(student["name"])
```

Dictionary Operation:

1. **Dictionary_name. key()**

   Return all the key of Dictionary

2. **Add new key-value pair:**

   Dictionary_name [“new_key”] =”value”

3. **Dictionary_name. value()**

Return all value of the dictionary

4. **Dictionary_name. update("key": "value")**

   Update and add new key-value pair to dictionary

5. **Dictionary_name. Items()**

   Return or print the key-value pair of the dictionary

6. **Dictionary_name. get("key")**

Example #04.3

```
# Example #03 Dictionary operation
student={
    "name":"muhammd haroon",
    "Roll_num":35,
    "marks":431
}
print(student.keys())  #Return all the key

print(student.values())     #Return all the value

student["ph#"]="0312834078"   #add new key:value pair

student.update({"name" :'Aqeel'})   #udate the value of specific key

print(student.items())      #Return or print the key-value pair of the dictionary

print(student.get("name"))    #Return the value of key that pass in function call
print(student)
```

Set in Python:

Set in python is a type of data structure that sorted collection of unordered and unique set of value.

Syntax:

Set are immutable meant the individual value are not change but the new element can be add.

Ignore the duplicate value.

Also support sting value.

The sequence of element are change after each execution.

Set_name= {1, 2, 3, 5, 4}

For empty set

Syntax:

Set_name= ()

Example #04.4

```
# Example 04 set
num={1,2,1,3,4,5}
print(num)      #duplicate value are ignored
```

Set operation and method:

1. **Set_name. add(el)**

   Add new element in set.

2. **Set_name. remove(el)**

   Remove the element from the set that are pass in the argument.

3. **Set_name. clear()**

   Delete the all element of the set

4. **Set_name. pop()**

   Remove the first element of set

5. **Set_name. union(second_set)**
6. **Set_name. intersection(second_set)**

Both union and intersection function give a new set the original set does not change.

Example 04.5:

```
# Example 05 set method
num={1,7,1,2,4,5}
num.add(6)      #add new element
print(num)

num.remove(1)       #remove the element
```

```
print(num)

# num.clear()     #delete all element of set
print(num)

print(num.pop())
print(num)

num2={2,6,7,5,9}
num3=num.union(num2)
print(num3)

num4=num.intersection(num2)
print(num4)
```

Lecture 04 code

Lecture #05

Topic:

1. **Loops**

Loops:

Loop are used to repeat a particular task or single instruction with a specific condition.

Types:

1. While
2. For

While Loop:

While loop is used when we does not known the exact number of iteration, they run as long as the condition is true.

Syntax:

while (condition):

#some work

Does not used bracket {} like c++ or JavaScript instead user proper spacing to make a bock of code.

Example 05.1

```
# Example 01 while loop
num =10
while(True):
    if(num==5):
        break
    print(num)   # 10 9 8 7 6
    num=num-1
```

For Loop:

They are the sequential traversal many use to traverse the list, tuple and set. Sequential traversing mean we know the actual number of iteration a loop can run.

Syntax:

for el in list_name:

#some work

Else

#some work

Else are completely optional and they will not execute if the loop is break using break keyword.

Else part execute with the loop end

Example #05.2:

```
#Example 02 for loop
num=[23,45,23,67,45,32,12]
count=0
for el in num:
    print(count,":", el)
    count+=1
print("End")
```

Range function:

The range () function give a sorted sequence of number by default starting from 0 to n and increase by one.

Syntax:

Range (start (optional), stop, step (optional)

Incremental condition

Range function number does not print directly. When we print in we get there initializing mean we get there definition statement

Example #05.3:

```
# Example 03 range function
num=range(1,5,2)
print(num)
for el in num:
    print(el)
print('The End')
```

Pass statement:

Pass is the null statement that identify the null statement in the loop and if-else.

Syntax:

For I in range (2)

pass

Example 05.4:

```
for el in range(2):
    pass
```

Break and Continue:

Break:

Break the loop after the particular condition

Continue:

Continue is use to skip a particular iteration and start the next iteration.

Syntax:

For I in range (5)

if (I==4):

break

if (i%2==0):

continue

print(I)

Example #05.5:

```
# Example 05 break and continue
num=0
while(True):
    num+=1
    if(num%2!=0):
        continue
    if(num>100):
        break
    print(num)
print("Value of num where break ", num)  #102
```

Lecture 05 code

Lecture #06

Topic:

1. Function
2. Recursive function

Function:

Function in python are the block of code that perform a particular specific task when call. They take argument and return a value, and they.

Advantages:

1. Remove redundancy
2. Readable code
3. Manageable

One time definition and any time we just call a function and they perform a desire takes.

Function Definition:

Use def keyword to define the function followed by a name and parameter in parentheses.

May function return some value or not

For return a value form function sample used the return keyword to return value

Syntax:

def function_name (parameter1, parameter2)

Function Call

Use function name followed by the argument in parentheses.

Syntax:

function_name (argument1, argument2)

Example #06.1:

```
#function definition
def sum(a,b):
    return a+b
#function call
print('the sum of two number is= ',sum(3,4))
```

Default parameter:

Value a function use when they call with no argument.

Syntax:

Function_name ()

Example #06.2

```
# Example 02 function with default parameter
def minus(a=1, b=0):
    return a-b
print("The subtraction of two number = ", minus())  #1
```

Type of function:

1. Built-in function
2. User define function

Built-in function:

They are the pre-define function like print (), len (), etc.

Example:

1. print ()
2. len()
3. type()
4. int()
5. float()
6. str()
7. range()

User Define Function:

They are function that is create by user by using def keyword

Recursive Function:

Recursive function is type of function that call itself to perform a repeated task until the base condition is meat.

Example #06.3

```
# Example 03 recursive function
def factorial(n):
```

```
    if(n==1):
        return 1
    return n*factorial(n-1)

print('The factorial of number is = ', factorial(5))        #120
```

Lecture #06 Code

Lecture #07:

Topic:

- File input/output

File:

File in python is use so stored the necessary data for restart next time form the same position.

File I/O:

Python can be used to perform operation on a file (write, read data).

Type of file:

1. Text file (txt, docx, .log)
2. Binary file(mp4, png, jpg)

File Operation:

Open a file:

Open function is used to open a file with the required mode if file not exist the will create.

Syntax:

File_pointer=open (“file_name.txt”, mode)

Mode of File Opening:

File Mode	Description
'r'	Read mode: They are default mode of file opening. They open file for reading
'w'	Write mode: They will open file in write mode and truncates (mean delete all data) existing file. Create file if not exist.
'a'	Append mode: They are write mode that does not truncate the existing file and add new data at the end
't'	Text mode: They are default mode. Use with other mode like rt, wt etc.
'b'	Binary mode: Used for binary file. Basically similar usage as the text mode like rb, wb, etc
'+'	Both mode: + mode is used to open a file for both read and write, and for write like r+, w+. And a+ open the file in append mode mean content does not loss.

Close file:

Close function is used to simply close the file

Syntax:

File_pointer. Close ()

Read a file:

For read data form a file we used read () function before reading we open the file in read mode (r) and the start reading

All_Content=file_pointer. Read ()

Read () function get complete data for the file in the string form.

We also mention the number of character that I supposed to read from the file like read (8). They read 8 character form the start

Example #07.1

```
# Example #01 file handleing basic
file=open("sample.txt", 'r')
data=file.read()
print(data)
file.close()
```

For line by Line

For reading file line by line we use getline () function for this purpose.

Syntax:

Line =file_pointer. getline ()

If the data is read form the file using getline() the file pointer is reach at the end of file so for next reading we first closed the file and then again open for perform reading operation from the start

Example #07.2

```
#Example #02 Line by line reading
file =open("sample.txt", 'r')
while True:
    data=file.readline()
    if not data:
        break
    print(data.strip())
file.close()
```

Write:

Write function is used to write a data into a file and for writing we open a file in write mode (w).

Syntax:

File_pointer=write (“data that write into file”, variable)

Example #07.3:

```
f=open("sample.txt", 'w')
f.write("I am student of computer science \n")
f.write("I Studey in University of Ajk \nI am form muzaffarabad \n")
f.write("Pakistan")
f.close
file =open("sample.txt", 'r')
while True:
    data=file.readline()
    if not data:
        break
    print(data.strip())
file.close()
```

With syntax:

In python **with** syntax is use for context management (like close file). It insure the resources cleaned up after the use.

Syntax:

With open ("file_name.txt", mode) as file_pointer

#block of code

#close file automatically

Alias:

In python, an alias is a nickname or alternate name for an object.

Example #07.4:

```
#Example #03: with syntax:
with open("sample.txt", 'r')as f:
    data=f.readline()
    while data:
        print(data.strip())
        data=f.readline()
```

```
        #Automatically close file
```

Delete a file:

Python does not provide direct method to delete file we delete file by import library called OS.

Example #07.5:

```
import os
os.remove("sample.txt")
```

Importing in python:

Import are the keyword that is used to bring external module (libraries) into your python script.

Syntax:

Import os

os.remove(“file_name”)

As os libraries are already exist

For install the new libraries like TensoFlow we used the following syntax for install:

Pip install tesnoflow

Pip3 install tensoflow

Type this in command prompt cmd for install

Lecture #07 code

Lecture #08:

Topic:

Object Oriented Programming

1. Basic concept
2. Abstraction

3. Encapsulation

Object-Oriented:

Object-Oriented Programming is a type of programming that revolves around “object” that have:

- Properties (data)
- Method (function that operator on that data)

In simulates real-world things (like cars, users, etc.) in code

Class & Object:

Class:

A class is like a blueprint/template for creating object. In class we define properties and method an object have.

Syntax:

```
#For creating a class
class class_name
    #block of code
#For creating an object:
class_name object_name ()
```

Object:

Object are an instance of a class. Has actual value and can use class method.

Attributes:

Attribute are the pieces of data associated with an object or class.

Type of Attributes:

1. Class Attributes
2. Object Attributes

The attribute who value is change for each object we called them object attribute and the attribute who value is constant for each object we called them class object.

Syntax:

class class_name

attribute_name=value

Accessing an Attributes:

print (object_name.attribute/method)

Method:

Method is a function that define inside a class, that operating on the object data.

Syntax:

class class_name

def function_name (self)

#code

Self-parameter are compulsive for define a function

Accessing a Method:

print (object_name.attribute/method)

Self-Parameter:

In python self refers to the instance of class. It's used to access the attribute and method.

- Self is the first parameter in method or function
- It represent the object calling the method

Example #08.1

```
#Example #01 basic syntax
```

```python
#Class Definition
class student:
    name="Haroon "
    def addsecondName(self, lastname):
        self.name+=lastname
#creating an object
s1=student()
s1.addsecondName("Rana")
print(s1.name)
```

Constructor:

In python, a constructor is a special method __init__ that initializes an object attributes when it's created. They function is automatically call when object is create.

Type of Constructor:

1. Default constructor
2. Parametrize constructor

The constructor that take no parameter is called default parametrize constructor.

That that parametrize are called parametrize constructor.

Syntax:

class class_name:

def __init__ (self)

#initializes the attributes

Example #08.2:

```python
#Example #02 constructor
class student:
    def __init__(self, name , marks, rollnumber):
        self.name=name
        self.marks=marks
        self.rollnumber=rollnumber
```

```
    def PrintInfo(self):
        print("Name \t\t\t RollNO. \t Marks")
        print(self.name,"\t" ,self.rollnumber, '\t',self.marks)
s1=student("Muhammad Haroon", [56,78,95], 35)
s1.PrintInfo()
```

Static Method:

Method that does not use self-parameter (work as class level)

Syntax:

@staticmethod

Def function_name() :

#block of code

Example #08.3:

```
# Example #03 static_method
class student:
    name="Muhammad "
    def __init__(self, name , marks, rollnumber):
        self.name+=name
        self.marks=marks
        self.rollnumber=rollnumber
    @staticmethod
    def getCollege():
        print("University Of AJK")
s1=student("Ali", 45, 90)
student.getCollege()
print(student.name)     #class attribute
print(s1.name, s1.marks)    #object attributes
```

Abstraction:

Hiding the implementation details of a class only showing the essential feature of the class.

Encapsulation:

Wrapping data and function into a single units (object).

Lecture #08 code

Lecture #09:

Topic:

1. *Advance Concept*
2. *Inheritance*
3. *polymorphism*

del keyword:

del keyword is used to delete the object or an attribute/properties of a class.

Syntax:

#for deleting attribute of object

del object_name.Attribute_name

#for deleting object of class

del object_name

Example #09.1:

```
class student:
    def __init__ (self, name, marks, RollNO):
        self.name=name
        self.marks=marks
        self.RollNO=RollNO
s1=student("Muhammad Haroon", 421, 35)
del s1.marks
#The name attribute was delete now we can not access
```

```
print(s1.marks)  #give an error

del s1
#So now stusent s1 object was delete we can not access and use
print(s1.name)   #Give an error
```

Private Attribute and Method:

In python, prefix with __ (double underscore) are used to make the attribute or method private.

Syntax:

#for making attribute as private attribute.

__ Attribute_Name__

#for making method as private method

__Method_name__

Private method and attribute does not access out of the class. Only used and access within the class

Example #09.2:

```
# Example 02 private attribute
class Account:
    def __init__(self, CN, Password):
        self.CN=CN
        self.__Password__=Password
    def changePassword(self):
        old=str(input("Enter the old password "))
        if(old==self.__Password__):
            new=""
            new=str(input("Enter the new password "))
            self.__Password__=new
        else:
            print("Enter the correct password ")
    def getPassword(self):
        return self.__Password__
A1=Account("Muhammad Haroon", 'haroon.15')
```

```
# print(Account.__Password__) #give an error because we going to access the
private attribute out of the class
A1.changePassword()
print(A1.getPassword())
```

Inheritance:

Inheritance mean drive another class that inherit the attribute and method from another base or parent class.

Syntax:

class ClassA:

#code

class ClassB (ClassA):

#code

Example #09.3:

```
# Example 3 Inheritance
class person:
    def __init__ (self ,name):
        self.name=name
class stusent(person):
    def __init__(self, roll,name):
        super().__init__(name)
        self.rollnumber=roll
s1=stusent(45,'Haroon')
print(s1.name)
```

class Base_class(Parient_class):

Type of Inheritance:

- *Single level*
- *Multi-level*
- *Multiple level*

Multi-Level:

```
#Example #4 Multi-level inheritance
class Animal:
    def __init__ (self , name):
        self.name=name
class Mammal(Animal):
    def __init__ (self, name):
        super().__init__(name)
class Cat(Mammal):
    def __init__ (self ,name, sound):
        self.sound=sound
        super().__init__(name)

c1 =Cat("Lion","Roar")
print(c1.sound)
```

Multiple Inheritance:

```
#Example #5 Multiple inheritance
class A :
    def __init__(self, A):
        self.A=A
class B:
    def __init__ (self, B):
        self.B=B
class C (A,B):
    def __init__(self, A, B,C):
        super().__init__(A)
        super().__init__(B)
        self.C=C
c1=C(1,2,3)
print(c1.A)     #2
```

Super Method:

Supper method is use to access the method of parent class.

Syntax:

super.function_name ()

Class Method:

A class method is bound to class and receives class as an implicit first argument. Used to change the class attribute

Syntax:

@classmethod

def function_name (cls):

#code

Also change by using this syntax:

Parent_class.Attribute

Note:

Static method are use where we does not need to access the parent attribute and class method are used to change or access the class attribute.

Example #09.4

```
class Student:
    name="Unknow"
    # def ChangeName(self ,name):
    #     Student.name=name
    @classmethod
    def ChangeName(cls,name):
        cls.name=name
s1=Student()
s1.ChangeName("haroon")
print(s1.name)
print(Student.name)
```

Type of Function:

1. *Static function ()*
2. *Normal function (self)*
3. *Class function (cls)*

Properties Decorator:

We uses properties decorator as the method in a class to use this method as the properties of the class.

We used *@properties* decorator when the attribute value is change when another parameter is change. So for tracking the change we use that function in the form of method called properties method.

1. *Setter*
2. *Getter*

Syntax:

@properties

def attribute_name (parameter):

#code

Example #09.5

```
# Example #05 Properties Dectorator
class Student:
    def __init__(self, name, math, phy, chem):
        self.name=name
        self.math=math
        self.phy=phy
        self.chem=chem
    @property
    def percentage(self):
        return (self.math+self.phy+self.chem)/3

s1=Student("Haroon", 89,56,78)
print(s1.name,"\t",s1.percentage)
```

Polymorphism:

Polymorphism mean many form. Mean method behave differently based on context.

Type of Polymorphism:

1. *Run-time Polymorphism (function overloading)*
2. *Compile time polymorphism (operator overloading)*

Operator Overloading:

Define how operator work for new drive and custom class

Syntax:

For addition:

A+B mean A.__add__(B)

Same for subtraction (__sub__), multiplication (__mul__)

Example #09.6

```
# Example #06 operator overloading
class complex:
    def __init__(self, real, img):
        self.real=real
        self.img=img
    def __add__(self,c):
        real=self.real+c.real
        img=self.img+c.img
        return complex(real, img)
    def printNum(self):
        print(self.real,"i +",self.img)
c1=complex (8,4)
c2=complex (3,4)
```

```
c3=c1+c2
c3.printNum()
```

Lecture #09 code