DAY 1 :INTRODUCTION TO PYTHON

- What is python?
  It is a easy,high level interpreted programming language known for its versatility and it uses indentation in its syntax

- Applications of python
- Data and analysis and vizualization
- Automation
- Web development
- Artificial intelligence
- Scientific computing
- Embedded systems

VARIABLES AND DATA TYPES
X=10#integer
x=2.34 # float
Name = “john” #string
Example: write a code to perform arithmetic operations

```
=int(input("enter a"))
b= int (input("enter b"))
print("addition=a+b",a+b)
print("multiplication=a*b",a*b)
print("difference=a-b",a-b)
print("division=a/b",a/b)
```

enter a2
enter b3
addition=a+b 5
multiplication=a*b 6
difference=a-b -1
division=a/b 0.6666666666666666

In [8]:

```
Ex:write a code to find area of circle
r=float (input("enter radius of circle"))
area= 3.142*r*r
print("area of circle",3.142*r*r)
```

enter radius of circle32
area of circle 3217.408

```
farenhit=float(input("enter the temperature in
farenhit"))
celsius=5/9*(farenhit-32)
print("temperature in celsius",celsius)
```

enter the temperature in farenhit21
temperature in celsius -6.111111111111112

DAY-2
CONTROL FLOW AND LOOPS

Day-2
Control flo1w and loops
If condition
syntax;
If condition:
Example;
If age==18

print(eligible for voting”)

If else
Syntax;

```
if condition:
    # block of code if the condition is True
else:
    # block of code if the condition is False
```

Example

```
If age==18:
print(eligible for voting”)
else:
print(“not eligible to vote”)
```

Elif loop;
syntax

```
if condition1:
    # Code to execute if condition1 is true
elif condition2:
    # Code to execute if condition is true (and condition1 is
false)
else:
    # Code to execute if neither condition1 nor condition2 is
true
```

Example

```
x = 10
if x > 20:
    print("x is greater than 20")
```

```
elif x > 5:
    print("x is greater than 5 but less than or equal to 20")
else:
    print("x is 5 or less")
```

Elif multiple statements can be done
Else if can be written in c but in python we write elif

For loop

```
for variable in iterable:
    # code block to be executed
```

Syntax;

```
for item in sequence:
    # Code block to execute
```

Example;

```
fruits = ["apple", "banana", "cherry"]
for fruit in fruits:
    print(fruit)
```

While loop
Syntax;

```
while condition:
    # code block to be executed
```

It depends on condition.
Example;

```
count = 0
while count < 5:
    print(count)
```

```
    count += 1
```

CONTROL FLOW TOOLS

Break

continue

Pass

1.Break

syntax

```
for variable in iterable:
    # Some code
    if condition:
        break  # Exit the loop
```

Example;

```
count = 0
while count < 10:
    if count == 5:
        break  # Exit the loop when count equals 5
    print(count)
    count += 1
```

The remaining pass and continue is the same.

Ex: write a code to find grade of student using if,elif,else statements

```
marks = int (input("enter the marks:"))
if marks>=90:
   print("grade:A")
elif marks>=75 :
   print("grade:b")
elif marks>=60:
   print("grade:c")
else:
   print("grade: F")
```

enter the marks:75

grade:b

Ex2: to find even or odd

```
num=int(input("enter the number:"))
if num%2==0:
    print("even numner:")
else :
  print("odd number:")
```

enter the number:46
even numner:

In [12]:

Ex: write a program to find sum of first n numbers

```
a=int(input("enter start range:"))
b= int (input("enter end range:"))
sum=0
for i in range(a,b+1):
  sum=sum+i
  print("sum of first n number",sum)
```

enter start range:4
enter end range:5
sum of first n number 4
sum of first n number 9

In [14]:

```
Ex: to find leap year
year = int(input("enter the year:"))
if (year%4==0 and year%100!=0) or (year%400==0):
  print("leap year:")
else:
  print("not a leap year:")
```

enter the year:2026

not a leap year:

In [18]:

```
Ex: substraction operatioaddition andn using
if and elif
a=int(input("enter the first number:"))
b=int(input("enter the second number:"))
operation= input("enter the operation")
if  operation == '+':
  print("addition",a+b)
elif operation == '-':
  print("substraction",a-b)
```

enter the first number:2
enter the second number:4
enter the operation+
addition 6

In [25]:

```
Ex:check positivity of number
a=int(input("enter the number:"))
if a>0:
  print("positive number")
elif a<0:
  print("negative numner")
```

enter the number:2
positive number

In [27]:

```
 Ex : write a program to find multiplication
table
a= int(input("enter the number:"))
for i in range(1,11):
   print(a,"*",i,"=",a*i)
```

enter the number:10
10 * 1 = 10
10 * 2 = 20
10 * 3 = 30
10 * 4 = 40
10 * 5 = 50
10 * 6 = 60
10 * 7 = 70
10 * 8 = 80
10 * 9 = 90
10*10=100

DAY 3 AND DAY4

FUNCTIONS AND MODULES

What is a Function?

- A function is a block of reusable code that performs a specific task. - Functions help make programs modular and easier to maintain.

Defining and Calling Functions

Syntax :
def function_name(parameters):
Example : def greet(name): print("Hello, " + name + "!")
greet("Alice")

Function Arguments

Positional Arguments:

def add(a, b):
return a + b
print(add(5, 3))

Keyword Arguments:

def greet(name, message):
print(message + ", " + name + "!") greet(name="Alice",
message="Good morning")

Default Arguments :

def greet(name, message="Hello"): print(message + ", " +
name + "!")
greet("Alice")
greet("Bob", "Hi")

Variable length arguments:

def sum_number(*numbers):
return sum(numbers)
print(sum_numbers(1,2,3,4)) # out put:10

Rerturn Statment:
Function can return values using the ‘return’ statment,

Example:
Def squar(num):
Return num*num

result=square(4)
print(“Squar is:”result)

MODULES:

A module is a file containing python code (functions,classes,variablers) that can be reused in other programs.

Using Built-in module:
- Important modules
- Import Specific Functions
- Renaming Module

- Write a program to print hello python?

```
def  show():
   print("hello python:")
show()
```

- Write a program to find the sum?

```
def sum():
```

```
  a = int(input("enter a number:"))
  b= int (input("enter a number"))
  sum = a+b
  print("the sum is :",sum)
sum()
```

- Write a program to find largest numbers?

```
def display():
    a = int (input("enter a number:"))
    b= int (input("enter a number:"))
    c= int(input("enter a number:"))
    if a>b and a>c:
      print("a is largest")
    elif b>a and b>c:
      print("b is largest")
    else:
      print("c is largest")
display()
```

- Write a program to find palindrome?

```
def palindrome():
  a = input("enter a number:")
  if a==a[::-1]:
    print("palindrome:")
  else:
    print("not palindrome:")
palindrome()
```

- Write a program to find armstrong numbers?

```
def armstrong():
  a = int(input("enter a number:"))
  sum=0
  temp=a
  while temp>a:
   sum= sum+((temp%10)**3)
   temp= temp//10
  if a==sum:
    print("it is a armstrong number:")
  else:
    print("it is not a palindrome:")
armstrong()
```

- Write a program find power of functions?

```
def power():
  a = int(input("enter a number:"))
  b = int (input("enter a number:"))
  print(a**b)
power()
```

- Write to program to find simple calculator?

```
def calculator():
  a = int (input("enter  number:"))
  b= int(input("enter number:"))
operation = input("enter  operation:")
if operation =='+':
      print("addition:")
      print(a+b)
```

```
elif operation =='-':
      print("substraction:")
      print(a-b)
elif operation == '*' :
      print("multiplication:")
      print(a*b)
elif operation =='/':
      print("division:")
      print(a/b)
else:
      print("invalid operation:")
calculator()
```

- Write a program to findeven or odd?

```
def even_or_odd():
  a = int(input("enter  number:"))
  if a%2==0:
     print("even:")
  else :
     print("odd:")
even_or_odd()

# reversed string
def reversed_string():
  a= ("enter  string:")
  print(a[::-1])
```

```
reversed_string()

#sum of digits
def sum():
  a = int (input("enter number:"))
  b = int (input("enter number:"))
  sum = a+b
  print("sum of numbers:", sum)
sum()

# default argument
def greef(name= "guest"):
 print(f"hello,{name}")
greef()
greef("syeda")

# factorial
def factorial():
  a = int (input("enter number:"))
  fact=1
  for i in range(a,a+1):
    fact = fact*i
    print("factorial",fact)
factorial()

#prime number check
def prime():
  num = int (input("enter number"))
```

```
  if num%5==0:
    print("it is a prime:")
  else:
    print("it is not a prime:")
prime()

#key word arguments
def ct ( colour , name):
    print(f"the person({colour}),is ({name})")
ct(colour = "blue",name = "syeda")

#using math module
import math
print(math.sqrt(44))
print(math. factorial(12))

# using random module
import random
print(random.randint(1,100))
print(random.choice([1,2,3,4,5]))

# gcd

def gcd(a, b):
    while b==0:
        a, b = b, a % b
    return a
    num1 = int (input("enter number:"))
```

```
num2= int(input("enter number:"))
result
```

Date Sturctures in Python:

LISTS
A list is an ordered,mutable collection of items.

List Operation:
- Accessing Items
- Modifying Items
- Adding Items
- Removing Items

Other Operation:
- Iterating Through a List

TUPLES:
A tuples is an ordered,immutable collection of items.
- Creating a tuple
- Accessing Items in a Tuple

DICTIONARIES:
A dictionaries is a collection of key-value pairs.
- Creating a dictionaries
- Accessing and Modifying Items

SETS:
A set is a unordered collection of unique items.

#list creation

```
name = [1.2,3,4,5]
print(name)

# accesing
print(name[0])

print (name[3])

print(name[2])

# create a list and add an element at the end and another at a
specific index
name=[1,2,3,4]
print(name)
name.append (3)
print(name)
name. insert(2,3)
print(name)

# list removal
name . remove(3)
print(name)
```

```
# list removal using pop
numbers=[1,2,3,2]
numbers.pop(2)
print(numbers)

# list sorting
name=[6,1,4,5]
name. sort()
print(name)



# list reversal
name=[2,5,6,8]
name.reverse()
print(name)

# sum of list elements
numbers=[1,2,3,4]
print (sum(numbers))

# maximum and minimum function
ball=[2,4,6,8,10,12]
print(max(ball))
print(min(ball))

# count occurences
mountain = [1,2,2,3,4]
```

```
print(mountain. count(2))
```

- Write a program tuple creation and access by using indexing

```
ocean =( 1,2,3,4,5)
print(ocean)
print(ocean[2])
```

Write dictionary creation

```
ocean = {
    "name": "indianocean",
    "depth":"5000",
    "colour": "blue"

}
print(ocean)
```

Write a program accesing dictionary value by using key value

```
ocean={
    "name": "indianocean",
    "depth": "5000",
    "colour" :"blue"
}
for key, value in ocean. items():
  print(f"indianocean,blue")
```

- updating elements from dictionary

```
asima = {
```

```
    "age": "27",
    "height":"5.6"

}
asima["age"]= "30"
asima["height"]= "5.6"
print(asima)
```

- addding elements from dictionary

```
asima = {
    "age": "27",
    "height":"5.6"

}
asima["age"]= "27"
asima["height"]= "5.6"
asima["hoppy"]= "hockey"
print(asima)
```

- iterating through a dictionary

```
ocean = {
   "name": "indianocean",
   "depth":"5000",
   "colour" : "blue"
   }
for key,value in ocean . items():
    print(key , value)
```

- removing elements from dictionary

```
ocean = {
   "name": "indianocean",
   "depth":"5000",
   "colour" : "blue"
   }
del ocean["colour"]
print(ocean)
```

DAY : 5

Introduction to Object-Oriented programming

OOP

A programming parading based on the consept of objects

- **Class **: A bluprint for creating objects
- **object**:An instance of a clasa.
- **Attributer**:Variables that hold data within a class.
- **Methodes**:Function defined in a class to perform action.

ENCAPSULATION:

Bunding of data and methods into a single unit (class).

INHERITANCE

A way to creat a new class using the features of an existing class.

- Single Inheritance
- Multiple inheritance

POLYMORPHISM:

The ability to redefind methodes in derived classes.

```
# encapsulation
class syeda:
  def __init__(self):
    self.public = "brillant"
    self.__private="fantastic"
  def asima__private(self):
    print(self.__private)
asima= syeda()
print(asima.public)
asima.asima__private()
```

```
polymorphism
lass shape:
  def area(self):
    passclass shape:

# polymorphism
  def area(self):
    pass
class Rectangle(shape):

  def __init__(self,length,width):
        self.length= length
        self.width = width
  def area (self):
    return self.length*self.width

class circle(shape):

  def __init__(self,radius):
    self.radius = radius

  def area (self):
    return 3.142*self.radius*self.radius

shape = [Rectangle(3,4),circle(6)]
```

```
for shapes in shape:
   print ("area",shapes.area())
```

```
area 12
area 113.112
```

```
class Point:
  def __init__(self, x, y):
      self.x = x
      self.y = y

  def __add__(self, other):
      return Point(self.x + other.x, self.y + other.y)

  def __str__(self):
      return f"({self.x}, {self.y})"

p1 = Point(2, 3)
p2 = Point(4, 5)
result = p1 + p2
print(result)
```