21 08 23

Python is very fokular language. It is used in various field including Software development, web development, computing, Big data artificial Intelligence. The interpreter is also called Python Shell. This symbol '>>>' is called python prompt which indicates that the interpreter is ready to recieve instructions. We can type commands or statement on this prompt for execution modes.

There are two way to run a program using the python interpreter

1) Interactive mode (>>>)
2) Script mode

1. Interactive mode :- In the interactive mode we can type a python statement on the >>> (prompt) directly. As soon as we press enter the interpreter executes the statements and display the result.

Working in the interactive mode is convinient for testing a single line code for instant execution but in the interaitive mode we can't save the statements for future use.

2. Sciptive mode :- In the sciptive mode we can write a python program in a file, save it and then use the interpreter to execute the program from the file such program file has .py extension and they are also known as script. Python scripts can be created using any editor. Python has a built in editor called IDLE which can be used to create program

Keywords :- Keywords are reserve words each word has specific meaning to the python interpreter ex:- True, false, class, finally, None, break, continue, and, def, del, is, an, except, import, return, assert, else, elif, del, for, from, global, if, import, in, lambda, pass, in, raise, try, while, with yield, or, not, nonlocal, assert.

Identifier :- In programming languages identifiers are named used to identify a variable, function, or other entity in the program. The rule for naming an Identifier are as follows :-

↳ The name should begin with upper case or lower case alphabet or underscore sign. These may be followed by any combination of character (a-z) (A-Z) (0-9) or underscore. It means identifier can't start with a digit

↳ It can be any length

↳ It should not be a Keyword or reserve word given in Keyword table

#Variable :- A variable is an identifier whose value can be changed for ex- variable age can have different value for different person. Variable name should be unique in a program. Value of variable can be string, number or combination of alphanumeric or character. In python we can use an assignment statement to create a new variable and assign specific value to them

* Write a Program to find sum of 2 no.

```
num1 = 10
num2 = 20
result = num1 + num2
print (result)
```

Date 26/08/23 Page no:

Data types :- Data types identify the types of data which a variable can hold and operation can be performed on those data type

⇒ Data types in python.

↳ Numbers → integer → Boolean
→ floating point
→ complex

↳ Sequences → strings
→ Lists
→ Tuples

↳ Sets

↳ None

↳ Mappings → Dictionaries

→ Numbers :- Number data type stores numerical value only it is further classified into 3 different types int, float and complex

Type	Discription	Ex
int	integer number	-12, -3, 0, 132
float	float point no.	-2.04, 4.14
complex	complex	3+4i, 2-2i

EX :-
>>> Price = -19219
>>> type (Price)
<class 'float'>

→ Sequence :- A python sequence is an ordered collection of items where each item is indexed by an integer value
Types of sequence data are String, list and tuple

↳ String :- String is a group of characters These character may be alphabet, digit or special character including space. String values are enclosed either in single quotation marks, double quotation mark, and triple quotation mark
We can't perform numerical expression on string

↳ List :- List is a sequence of items separated by commas. An item's are enclosed in square bracket

EX :-
>>> List 1 = [5, 3.4, "New Delhi", "20c"]
>>> List 1
O/P :- [5, 3.4, "New Delhi", '20c', 4]

↳ Tuples :- Tuples is sequence of item separated by commas and items are enclosed in paranthesis. Once

created we can not change item in the tuple

Ex :-

```
>>> tuple1 = (5, 3.4, "New Delhi", "20c")
>>> print (tuple1)
```

O/p :-

```
(5, 3.4, 'New Delhi', '20c')
```

→ None :- Mapping :-

↳ Dictionary :- Dictionary in Python hold data item in key-value pairs and items are enclosed in curly brackets {}. Dictionaries permit faster aces to data. Every key is separated from its value using a colon (:) sign The key-value pairs of a dictionary can be accsed using key. Keys are usually of string type and their values can be of any data type. In order to acss any value in the dictionary, we have to specify its key in square bracket

ex :-

```
>>> dic1 = {'fruit' : 'Apple',
            'climate' : 'cold',
            'price(Kg)' : 120 }
>>> print (dict1)
```

o/p :-
{'fruit' : 'Apple', 'climate' : 'cold',
'Price(kg)' : 120}

# getting value by specifying a key
>>> print (dic['Price(kg)'])
o/p 120

Operators :- An operator is used to perform specific mathematical or logical operation on value. The value that the operator work are called operands
Types of operators are :-

1) Arithmatical operator :-
+, -, *, %, ** (exponent), // floor division

2) Relational operator :- ==, !=, <, > etc

3) Assignment operator :- =

4) Logical operator :- AND, OR, NOT

5) Membership operator :- Membership operator is used to check if a value is a member of given sequence or not, ex :-

- In :- returns true if the variable or value is found in the specified sequence and false otherwise

ex:- >>> num = [1, 2, 3]
>>> 2 in num
o/p :- true

- not in :- returns true if the variable or value is not found in the specified sequence and false otherwise

28/08/23

Set :- Set is unordered collection of items & separated by commas and the items are enclosed in curly bracket. A set is similar to list and except that it cannot have duplicate entry

ex:- >>> set1 = {10, 20, 3.14, "New Delhi"}
>>> Print (type(set1))
o/p <class 'set'>
>>> Print (Set1)
o/p :- {10, 20, 3.14, "New Delhi"}

None :- None is special data type with a single value it is used to signify the absence of value in a situation.

ex :- >>> MyVar = None
>>> Print (type (MyVar))
o/p :- <class 'NoneType'>
>>> Print (MyVar)

o/p :- None.

Mutable and Immutable Data type

Variables whose value can be changed after they are created and assigned are called Mutable.

ex :- List, set and Dictionary.

Variables whose value can not be changed after they are created and assigned are called Immutable

ex :- Integer, float, Boolean, Complex, strings, tuples

Identity operator :- Identity operators are used to determine whether the value of variable is of a certain type or not. Identity operator can also used to determine whether two variable are refering to the same object or not

⇒ Types of identity operator

- is
- not is

ex :- >>> num1 = 5
>>> type (num1) is int
True
>>> num2 = num1
>>> id (num1)

1433920576
>>> id (num2)
1433920576
>>> num1 is num2
True.

Preference of Operator

Order of Preference	Operator
1	**
2	+, -, ~ (not)
3	*, /, %, //
4	+, -
5	<=, <, >, >=, ==, !=
6	=, %=, /=, //=, -=, +=, *=, **=
7	is, is not
8	in, not in
9	not
10	and
11	or

Input and Output :-

⇒ Example for input :-

>>> fname = input ("Enter name")
>>> age = input ("Enter age")

>>> type (age)
o/p :- <class 'str'>
⇒ To take int. as input
>>> age = int(input("Enter age"))

⇒ Example of Output
print("Hello")

Predefined function :-
↳ abs() ↳ divmod() ↳ max()
↳ min() ↳ pow() ↳ sum()
↳ list() ↳ Tuple() ↳ set()
↳ bool() ↳ chr() ↳ dict()
↳ float() ↳ int and many more.

29/08/2023

If- else statement :-
There are situation where we have more than one option to choose from, based on the outcome of certain condition this can be done using if-else conditional statement
There are 3 ways to write if-else statement :-
↳ if statement :-
age = int(input("Enter age"))
if age >= 18 :
print("eligible to vote")

↳ If .. else :-

```
num1 = int(input("Enter first No."))
num2 = int(input("Enter second NO."))
if num1 > num2:
    diff = num2 - num2
else:
    diff = num2 - num1
print("The diff. of", num1, "and",
      num2, "is", diff)
```

↳ If ... elif ... else

Q1) Write a program to check whether no. is +ve, -ve or 0.

Soln

```
num. = int(input("Enter any no."))
if num > 0:
    print("+ve number")
elif num < 0:
    print("-ve number")
else:
    print("The number is 0")
```

Q2) WAP whether no. is even or odd

```
num = int(input("Enter any num"))
if num % 2 == 0:
    print("Number is even")
else:
    print("Number is odd")
```

Q3) Find the largest No. from 3 No.s

```
num1 = int(input("Enter first No."))
num2 = int(input("Enter second No."))
num3 = int(input("Enter 3rd No."))
Print("\n The numbers are : \n first No.:", num1, "\n Second No. :", num2, "\n Third No. :", num3)
if (a>b and a>c):
    Print("\n", num1, " is the greatest among three that is 1st number")
elif (b>a and b>c):
    Print("\n", num2, "is the greatest among three that is 2nd number")
elif (c>a and c>b):
    Print("\n", num3, " is the greatest among three that is 3rd number")
elif (a==b and a>c):
    Print("\n First number :", num1, "and Second Number :", num2, "are equal and greater than third number :", num3)
elif (b==c and b>a):
    Print("\n Second number :", num2, "and third number :", num3, "are equal and greater than first number :", num1)
elif (c==a and c>b):
    Print("\n first number :", num1, "and third number :", num3, "are equal and greater than second number :", num2)
```

Looping Statement

↳ for loop :-

```
for <control-variable> in
    <sequence/items in range>:
    <statement inside body of the loop>
```

Q1) Write a program to print even no. from the given sequence

numbers = [1,2,3,4,5,6,7,8,9,10]

```
for num in numbers:
    if num%2 ==0:
        print(num, "is an even no.")
```

Q2) Wap to print the number in given sequence.

```
number = [1,2,3,4,5]
for num in numbers:
    print(num)
```

↳ while loop :-

```
while test_condition:
    body_of_while
```

Q1) Program to print first 10 natural number.

```
i = 1
while (i<=10):
```

```
print (count)
count += 1
```

Q2) Program to find factorial
Q3) Program to check if input is prime or not
Q4)

```
# Program to find factorial
num = int(input("Enter the number:"))
fact = 1
if num < 0:
print("sorry factorial doesn't exists")
elif num == 0 or num == 1:
print("factorial of", num, "is 1")
else:
for i in range(1, num+1)
    fact = fact * i
print("The factorial of", num, "is :", fact)
```

```
# Program to check if input is prime or not
num = int(input("Enter the number:"))
flag = False
if num == 1:
print(num, "is not a prime number")

elif num > 1:
for i in rang(2, num):
```

```
if (num % i) == 0:
    flage = True

if flag:
    print(num, "is not a prime number")

else: print (num, "is a prime number")
```

04/09/23

#Identation:- In most programming language the statement within a brackets are put inside a curly bracket.

However python uses indentation for block as well as for nisled block structure. Leading whitespace (space and tabs) at a begining of statement is called Indentation

#Break :- The break statement alters the normal flow of execution as it terminates the current loop and ~~xxxxxx~~ resume the execution of the statement following that loop

ex:-
```
num = 0
for num in range (10):
    num = num + 1
    if num == 8:
        break
    print ("Num has value" + str(num))
```

Print("encountered break !! out of loop")

Continue Statement :- when a continue statement encountered the control skips the execution remaining statement inside the body of loop for the current iteration and jumps to the begining of loop for the next iteration. If the loops condition is still true the loop is entered again, else the control is transffered to the statement immidiately following the loop

ex:-
```
num = 0
for num in range(6):
    num = num + 1
    if num == 3:
        continue
    Print("Num has value " + Str(num))
Print("end of loop")
```

Range function :- Range is built in function

Syntax of range function is

range ([START], stop, [STEP])

The range function returns a sequence of numbers starting from 0 by default and increment by 1 by default and stops before a specified number

→ Start :- (Optional). An integer no. specify at which position to start by default is 0

→ stop :- (required.) An integer no. specify at which position to start

→ Step :- (Optional). An integer no. specify the increment by default is 1

ex:- >>> list (range (10) -)
[0, 1, 2, ... 9]

>>> list (range (2, 10))
[2, 3, ... 9]

>>> list (range (0, 30, 5))
[0, 5, 10, 15, 20, 25]

step value is -1 Hence decreasing
# sequence is generated
>>> range (0, -9, -1)
[0, -1, ..., -7, -8]

ex:- num = int (input ("enter a no."))
for i in range (1, num + 1):
    for j in range (1, i+1):
        print (j, end = " ")
print()

Date 6/9/23

Pass :- Pass statement does nothing in python which is helpful for using a placeholder in if statement branches, function and classes.

Q) Print out the multiple of 7 from 1-71

Soln

```
for number in range(1,71):
    if number %7 != 0:
        pass
    else:
        print(f"{number} is multiple of 7")
```

List :- The data type list is an ordered sequence which is mutable and made up of one or more elements a list can have element of different data type such as integer, float, string, tuple or even another list. A list is very useful to group elements of mixed data type. Elements in list are enclosed in square brackets and separated by commas

ex :- >>> List = [2,4,6,8,10,12]
>>> print(List[1])

Each element in a list is accessed from index. The first index value is 0, second is 1 & so, on

ex >>> list[0]

List is mutable :-

In python lists are mutable that means the content of list can be changed after it has been created

ex :-

```
>>> List 1 = ['RED', 'Green', 'blue', 'Dem
>>> List 1 [3] = 'black'
>>> List 1
```

⇒ List Operation :-

→ Concatination :- Python allows us to join 2 or more list to using concatination

```
>>> List 1 = [1, 3, 5, 7, 9]
>>> List 2 = [2, 4, 6, 8, 10]
>>> List + List2
```

NOTE = There is no change in original List

→ Repeatition :- Python allows us to replicate the content of the list using repeatition operation.

```
>>> List 1 = ['Hello']
>>> list * 4
['Hello', 'Hello', 'Hello', 'Hello']
```

Date: / / Page no:

▶ Membership :- The membership operation 'in' checks if the element is present in the list returns true else returns false

ex:- >>> list1 = ['RED', 'GREEN', 'BLUE']
>>> 'GREEN' in List1

▶ Slice :- Slicing operation allow us to create new list by taking out from existing list

ex:- >>> list1 = ['RED', 'GREEN', 'BLUE', 'CYAN', 'Magenta', 'Yellow']
>>> list [2:5]
Included → excluded
>>> list [:5]

Traversing a list :-
We can access each element of a list or traverse a list using for loop or while loop

→ Using for loop

```
list1 = ['RED', 'GREEN', 'BLUE', 'YELLOW',
         'BLACK']
for item in list1:
    print(item)
```

Using for with range function

```
for i in range(len(list1)):
    print(list[i])
```

⇒ List methods and built-in functio

↳ len() :- It gives length

↳ list() :-

↳ append() :-

```
>>> list1 =[10,20,30,40]
>>> list1.append(50)
>>> list1
```

O/p :- [10,20,30,40,50]

↳ extend :-

```
>>>list1 =[10,20,30]
>>> list2 =[40,50]
>>> list1.extend(list2)
```

↳ insert() :-

```
>>> list1 = [10,20,30,40,50]
>>> list1.insert(2,19)
>>> list1
```

O/p :- [10,20,19,30,40,50]

↳ count() :-

```
>>> list1 =[10,20,19,19,19,40]
>>> list1.count(19)
```

O/p :- 3

↳ find ():-

>>> list1 = [10, 20, 30, 40, 50]

>>> list1.index(20)

o/p :- 1

↳ remove ():-

>>> list1 = [10, 20, 30, 40]

>>> list1.remove(10)

o/p :- [20, 30, 40]

↳ POP ():-

>>> list1 = [10, 20, 30, 40, 50]

>>> list1.POP(3) → index

o/p :- 40

↳ reverse ():-

>>> list1 = [34, 66, 12, 89, 28, 29]

>>> list1.reverse()

>>> list1.

o/p :- [99, 28, 89, 12, 66, 34)

↳ sort ():-

list1.sort()

↳ min () → max() ↳ sum()

Q) WAP to bottom any three list

operation given in menue.
① Append an element
② Insert an element
③ append a list to the given list
④ Modify an existing list
⑤ Delete an existing element from its position
⑥ Delete an existing element with a given value
⑦ Sort the list in the ascending order
⑧ Sort the list in descending order
⑨ Display the list

Tuple :- It is an ordered sequence of element of different data type as integer, float, string or even a tuple. Element in tuple are enclosed in paranthesis (or Round bracket) and are separated by commas.

Tuple is an immutable object which means it can not be changed and we use it to represent fixed collection of item. Indexing starts from 0
ex:- tuple1 = (20, 10, 20, 'Apple', 25.6)

↳ Concatination :- » tuple1 = (20, 10, 20, 30)
» tuple2 = (2, 4, 6, 8, 10)

>>> tuple1 + tuple2
o/p:- (20,10,20,30,30,2,4,6,8,10)

>>> tuple1 = tuple1 + (6,)
>>> tuple1
o/p:- (20,20,20,30,6)

→ Repeatition:- ~~Repeation~~ Repeatition operator is depicted by '*'. We can repeat the tuple item.
ex:- >>> tuple1 = ('Pawan')
>>> tuple1 * 3

→ Membership:- The 'in' operator checks if the element is present in tuple returns true else returns false
ex:- >>> tuple1=('GREEN', 'RED', 'BLACK')
>>> GREEN in tuple1

→ Slicing :- Like string and list slicing can be applied in tuple also
ex:- >>> tuple1 = (10,20,30,40,50)
>>> tuple[1:3]
o/p:- (20,30,40)

⇒ Tuple methods and built in function
→ len()
→ tuple() :- >>> tuple1 = tuple()

↳ Count():-

```
>>> tuple1 = (2, 19, 19, 20, )
>>> tuple1.count(19)
```

o/p:- 2.

↳ Index()

```
>>> tuple.index(20)
```

↳ max() ↳ min() ↳ sum()

↳ sort()

⇒ Nested Tuple:-

```
>>> st = ((101, 'AMAN', 98), (102, 'GEET', 95
          (103, 'SAHIL', 87))
print("S NO.", "Roll NO", "Name", "Marks")
for i in rang(0, len(st)):
    print((i+1), '\t', st[i][0], '\t',
          st[i][2], '\t', st[i][2])
```

Ques1) WAP to swap two number without using a temporary variab

Ques2) WAP to calculate the area and circumference of circle using a functio

Ques3) WAP to input n no.s from the user store these no. in tuple and print the max and min from these tuple

Program to swap two No. without using a temporary variable

```
a= int(input("Enter the value of a :"));
b= int(input("Enter the value of b:"));
print(f" Value of a and b before swapping
    :\n a={a}  b={b}")
a = a+b
b= a-b
a= a-b
print(f" Value of a and b after swapping
    :\n a={a}  b={b}")
```

Program to calculate Area and Circumference of Circle

```
import math
pi = math.pi
r= float(input("Enter the radius :"));
def circle (r):
A= pi*r*r
c= 2*pi*r
print(f"Area of circle: {A}\n Circumference
    of circle : {c}")
circle(r)
```

Program to check max and min in tuple

```
t= tuple(input("Enter the number by
    leaving space :").split())
```

print ("Tuple =", t)
print (f"Maximum element from the tuple is : {max(t)}\n Minimum element from the tuple is : {min(t)}")

26/09/2023

Dictionary :- The data type dictionary fall under mapping. It is a mapping b/w a set of keys and a set of values. The Key-Value pair is called an item. A Key is separate from its value by colon (:) and consecutive items are separate by commas

→ Creation of Dictionary :-
A dictionary is a collection of ordered unordered, changeable and do not allow duplicates.
Dictionaries are changeable means we can change, add, or remove item after the dictionary has been created
Dictionary can not have two items with the same key
ex:- >>> Dict1 = { } # empty dictionary
>>> dict2 = {'Mohan' : 95, 'Ram' : 89}

→ Access items in a dictionary :-
The items of the dictionary can access

via the Key rather than their relative position. Each Key serves as the index and maps to the value.
ex:- >>> dict2[Ram] o/p:- 89

→ Adding an element :-
>>> dict2['MEENA'] = 78
>>> dict 2 o/p:- all the value with newly added value

→ Modify the existing value :-
>>> dict2['MEENA] = 20 # Modified the
>>> dict 2 value of meena from 78 to 20

→ Membership operation:-
The membership operation 'in' checks if the Key is present in the memory returns True else return false
And 'Not in' operator returns if the Key is not present in the dictionary and else returns false
ex:- >>> Suhel in dict2
o/p:- True
>>> Pawan not in dict2
o/p:- True

→ Traversing a dictionary :-

ex:- >>> for key in dict2:
print (key, ':', dict2[key])

27/09/2022

⇒ 2nd way to use for loop

>>> for key, value in dict2.items()
print (key, ":", value)

☆ Dictionary methods and built in function

→ len() → dict():-

>>> pair1 = [('Mohan', 95), ('Ram', 89),
('Raj', 92)]

>>> pair1

>>> dict1 = dict(pair1)

>>> dict1

o/p :- {'Mohan': 95, 'Ram': 89, 'Raj': 92}

→ key() :- Returns a list of key in dictionary

ex:- >>> dict1.key()

→ values() :- It only shows value of dictiona

ex :- >>> dict1.value()

→ items() :- It is used print both key & value

→ get() :- >>> dict1.get(Mohan)

↳ update() :- Append the key value pairs of the dictionary passed as the argument to the key value pair of the given dictionary

ex:- >>> dict1 = {'Mohan' : 95, 'Ram' : 89, 'Raj' : 92}
>>> dict2 = {'Sohan' : 79, 'Geeta' : 89}
>>> dict1.update(dict2)
>>> dict1

↳ delete() :- delete the items with the given key ex:- >>> del dict1['Ram']

↳ clear() :- clears all the items of the dictionary
ex:- >>> dict1.clear()

03/10/2023

Python function
A function is a block of code which only runs when it called. You can pass data known as parameter into function. A function can return data as a result

⇒ Creating a function :- In python a function is defined using def keyword
Syntax :- def func_Name (Parameters):
Body of the function

ex:- def Myfunc():
print("Hello")

→ arguments:- Information can be passed into function as arguments Arguments are specified after the function name, inside the paranthe You can add as Many arguments as you want, just Separate them with a comma.

ex:- def Myfunc(fname):
print(f"Hello" {fname})

Myfunc("Pawan")

o/p:- Hello Pawan

- Arbitary arguments:- If you don't know how many arguments that will be passed into your function that * before parameter name in the function definition This way the fenction will recieve a tuple of argument and can access the item accordingly.

ex:- def Myfunc(*kids):
print("The youngest child is:" + kids[2])

Myfunc("Ram", "Mohan", "Seeta")

• Keyword argument :- You can also send argument with the key = value syntax
ex:- def My_func (child3, child2, child1):
print ("The Youngest child is :", child3)
My_func(child1 = "Ram", child2 = "Mohan" child3 = "Seeta")

This way the order of argument doesn't matter.

18/10/23

#Sets:- Unordered collection of items seperated by commas, enclosed in curly braces. Set is immutable it can take number, string, tuple

⇒ operation we can perform :-
↳ Union :- A.union(B) or A|B
↳ intersection :- A.intersection(B) or A&B
↳ difference :- A.difference(B) or A-B
↳ Symmetric difference :- A∩B ∘
A = {1, 2, 3, 4}, B = {2, 4, 6, 8}
A∩B = {1, 3, 6, 8}
A.symmetric_difference(B)

Built in function :-
① add() :- It will add the element to the specified set. ex:- A.add(5)

(2) update ():- A. update (B)

(3) remove ():- A. remove (4)
If 4 is not present in the set it will give exception

(4) discard ():- A. discard (4)
It will not give exception 4 is not present there

(5) POP ():- A.pop() it will give error when the set is empty

(6) isSubset ():- It will check whether the given set is a subset of other specified set or not.
A1. issubset (A2)

(7) isSuperset () (8) clear ()

Function,
There are two types of function
→ built in function → user difined

→ function without parameter & without return type
→ fenction with parameter & without return type
→ function with parameter & with return

types

→ function without parameter & with return type

⇒ Simple interest

```
def SI (P, r, t):
    s = (p*r*t)/100
return s

print (SI(10, 20, 3)),
```

→ default argument :-

```
def func (country, state = "MP"):
    print (country, state)

func("India")
```

Q) Create function for the following
① fabonacci series ② prime number
③ Leap year ④ Palindrome number
⑤ Sum of list element

Program to print fibonacci series

```
def fib(n):
    if (n==1 or n==2)
        return n-1
```

```
    else:
        return fib(n-1) + fib(n-2)

a = int(input("Enter the place value till
        which you want to print:"))
print(f"\n The value till {a} the places are:")
for i in range(1, a+1):
    print(fib(i), end = " ")
```

Program to check for prime number

```
def prime(n):
    flag = false
    if num == 1 or num == 0:
        print(num, "is not a prime no")
    else:
        for i in range(2, num):
            if (num % i == 0):
                flag = True
        if flag:
            print(num, "is not prime")
        else:
            print(num, "is prime")
num = int(input("Enter any number"))
prime(num)
```

Program to check for leap year

```
def leap_year(n):
    # condition for century years
```

```
if (year % 400 == 0 and year % 100 == 0):
    print (f"{year} is a leap year")
elif (year % 4 == 0 and year % 100 != 0):
    print (f"{year} is a leap leapyear")
# above condition is for non century year
else:
    print (f"{year} is not a leap year")
year = int (input ("Enter the year"))
leap_year (year)
```

Program to check if a given string is a palindrome or not

```
def Palindrome(str):
    flag = false
    for i in range (1, len(str)):
        if (str == str[::-1]):
            flag = True
    if flag == True :
        print (f"Given string is palindrome")
    else:
        print (f"Given string is not palindrome")
a = input ("Enter any string:")
Palindrome (a)
```

Program to print sum of element of a list

```
def sum (l):
    total = 0
```

```
    for ele in range( len(l)):
        total = total + l[ele]
    print(f" Sum = {total}")
l=[]
n=int(input("Enter the length of list :"))
for i in rang(n):
    num=int(input(f"Enter value at {i} inde
    l.append(num)
sum(l)
```

```
for ele in range(len(l)):
    total = total + l[ele]
print(f" Sum = {total} ")
l=[]
n=int(input("Enter the length of list :"))
for i in rang(n):
    num= int(input(f" Enter value at {i} index:"))
    l.append(num)
sum(l)
```

• Arbitary Keyword arguments:- When we don't know the order and how many arguments are to be passed in the function then we use this

Syntax :- **kwargs

```
ex :- def message (**kwarg):
        print(" fname = ", name[fname])
        print(" lname = ", name[lname])
        print(" mname = ", name[mname])

Message(fname = 'Yash Raj', mname = 'Singh'
        lnam = 'Chauhan')
```

Dictionary :- We can also create dictionary by dict() function

ex :- ① dict (1 = "Pawan", 2 = "Singh"
3 = "Bhadauriya")

Here we have passed Keyword argume

② We can also pass list of tuple where the length of tuple is 2

```
ex = l =[(1, Pawan), (2, aman)]
     dict(l)
```

→ Accessing dictionary :-

```
d={'age' =22, 'Weight' =30, name : {fname
  : 'Yash', mname : 'singh', lname : 'chouhan'}
  pet : [dog, cat, cow], float : 3.14}
```

```
>>> d[name][fname]   o/p - Yash
>>> d[pet][1]        o/p - cat
```

Program to count the occurance of a charachter in a string

```
def char_freq(str1):
    str = str1.lower()
    str2 = str2.replace(" ", "")
    dict = {}
    for n in str2:
        keys = dict.key()
        if n in keys:
            dict[n] = dict[n]+1
        else:
            dict[n] = 1
    return dict
print(char_freq("Pawan Singh Bhadauriya")
```

Date: 27/10/23 Page no:

#frozen set :- It is a set which is immutable. We can create this by frozenset() function.

ex = Vowels = {'a', 'e', 'i', 'o', 'u'}
B = frozenset(Vowels)

- When we want to put set in an another set as a value at that place we require frozen sets.
- We can also us frozen sets as a key in the dictionary as they are immutable

↳ Empty frozen set :- A = frozenset()

↳ Syntax :- frozenset([<iterables>])
parameter

ex :- dict1 = {1 : 'Pawan', 2 : 'Yash'}
A = frozenset(dict1)
print(A)
o/p :- frozenset({1, 2,})

↳ Operations that we can perform in frozen set

- copy() :- A = frozenset = ({1, 2, 3, 4})
C = A.copy()

Date: / / Page no:

↳ Union() :- It is same like sets
A = frozenset({1, 2, 3, 4})
B = frozenset({2, 4, 6, 8})
C = A.union(B)

↳ Intersection() :- Same as set

↳ difference()

↳ Symmetric_difference()

↳ Is_disjoint() :- A = frozenset({1, 2, 3, 4, 5, 6})
B = frozenset({5, 6})
A.isdisjoint(B)
it gives output as true or false

(Q) Take a list = [8, 16, 32, 64, 128, 256, 512, 1024]

(i) o/p :- [16, 8, 64, 32, 256, 128, 1024, 512]
(ii) o/p :- [16, 64, 256, 1024, 8, 32, 128, 512]

Ans (i) print("The list element must be even")
l = []
n = int(input("Enter the length of list:"))
for i in range(n):
ele = int(input(f"Enter the {i} element"))
l.append(ele)
print(l)

for i in range(len(l)):
if(i%2 == 0):

```
a = l[i]
l[i] = l[i+1]
l[i+1] = a
```

print("list after shuffeling", l)

02/11/23

t = tuple(num) this will give an error as num is not an iterable object

OOPS :-

When we create an object of the class then the memory will be generated

→ class :- collection of object
→ Object :- collection of properties & behaviours

⇒ OOP concepts
① classes ② object ③ Inheritance
④ Encapsulation ⑤ Abstraction ⑥ polymorphism
these are more concept rather than them
① Couplining ② Cohission ③ Agregation
④ specialization ⑤ Generalisation

⇒ classes in python :-

```
class Employee :
    id
    name
    Sal
    def __init__(self, id, name, sal):
        self.id = id
        self.name = name
        self.sal = sal
    def show_sal(self):
        print(self.sal)
E1 = Employee(1, 'Pawan', 5000)
E1.showsal()
Employee.showsal(E1)
```

ex:-
```
class dog :
    list1 = [ ]
    def __init__(self, breed)
        self.breed = breed
        self.list1.append(1)
    def prints(self):
        print(self.breed)
        print(list)
d1 = dog('German')
```

ex:- create a class student

```
Class student :
    name
    roll_no.
```

Course
address
Age
def __init__(self, name, age, roll_no, address, course):
    self.name = name
    self.roll_no = roll_no
    self.age = age
    self.address = address
    self.course = course
def showdetails(self):
    print("name :", self.name, "age", self.age, "course:", self.course, "address", self.address)

S1 = Student('Pawan', 19, 57, 'XYZ', 'MCA')
S1.showdetails()

Ex:- class student:
    self.name = 'Pawan'
    self.age = 19
    self.roll_no = 57
} These are class variable

    def fun():
        print("Hello")
x = fun()
Class student:
    f = x
    #Body

xf = student.f // Calling a function through class object

08/11/2023

Inheritance:-
ex:-
class Person:
    def __init__(self, fname, lname):
        self.fname = fname
        self.lname = lname
    def display():
        print(self.fname, self.lname)

class student(Person):
    def __init__(self, fname, lname, age):
        self.fname = fname
        self.lname = lname
        self.age = age
    person.__init__(self)
    def printgreet(self):
        print("Hello")

S1 = Student('Pawan', 'Bhadouriya', 50)

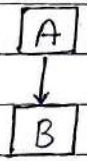
NOTE :- By default object class is inherited.

ex:- class Vehicle:
    def __init(self, V_no, V_color):
        self.V_no = V_no

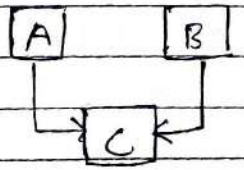
```
        self.v_color = v_color
    def display(self):
        print("Vehicle no. = ", v_no, "\n
              Vehicle color = ", v_color)

class Car(Vehicle):
    def __init__(self, m_name, company, kms):
        self.m_name = m_name
        self.company = company
        self.kms = kms

Vehicle.__init__(self)
def showdetails(self):
    print("m_name =", m_name, "\n
          company =", company, "\n no. of
          kms = ", kms)

C1 = Car('THAR', 'Mahindra', 200)
C1.display()
C1.showDetails()
```

22/11/2023

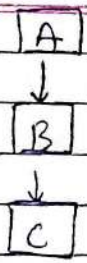
↳ Single Inheritance :-



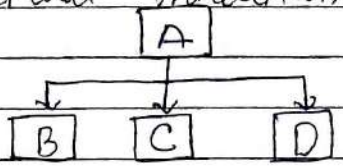
↳ Multiple Inheritance :-



↳ Multilevel Inheritance :-



↳ Hierarchical Inheritance :-



⇒ Method overloading :- Same name with different parameters

⇒ Method overriding :- Same name and parameter but different definition

```
ex:- class MethodDemo:
        def fun(self):
            print("fun")

class override(Methoddemo):
    def fun(self):
        print("This is overrid method")
        print("bye")
        super().fun()

obj = override()
obj.fun()
```

Date: / / Page no:

Data hiding :- To make any variable or Method private then we'll use double underscore ex :- __ variable

Errors :-
There are three types of error
→ Compile time → Run time
→ Logical

Exception :-
BaseException
→ BaseExceptionGroup
→ GeneratorExit
→ KeyboardInterrupt
→ SystemExit
→ Exception
→ Arithmetic Error
→ FloatingPointError
→ OverflowError
→ ZeroDivisionError
→ EOF error
→ Name Error
→ Type error
→ File NOT found Error
→ OS Error
→ Import Error
→ Buffer error
→ Syntax Error
→ IndentationError
→ Value Error
→ Warning

Exception Handling :-
Try, except, throw, finally

ex :-
```
a = 5
b = 0
try:
    c = a/b
except:
    print("error")
```

→ try with multiple except
```
try:
    for i in range(4):
        print(l[i])

except TypeError:
    print("TypeError")

except ZerodivisionError:
    print("zerodivision error")

finally:
    print("Hello")
```

⇒ try - except - else:

```
try:
    # exception Handling
except:
    # Handling exceptions
else:
    # if no exception then even this code
finally: # always even
```

⇒ try:

```
a = int(input("Enter first number:"))
b = int(input("Enter second number:"))
print(c)
except Exception as e:
    print(e)
```

⇒ built in exception:

```
a = int(input("Enter number:"))
if a > 5:
    raise exception("No. is greater than 5")
```

23/11/2023

file handling

↳ Open() :- Used to open a file

```
f = Open("Demo.txt", 'r')
```

↳ mode of file

↳ Read() :- Used to read a file

```
f.read()
```

↳ close() - used to close a file

```
f.close()
```

↳ write() :- To write in a file

```
f = open("Demo.txt", 'w')
f.write("Welcome to my world")
f.close
```

It will erase the old data. So if we didn't want this so we can open the file in append mode

ex:- `f = open("Demo.txt", 'a')`

↳ writelines() - It will write the data line by line, it write the list of string

ex:- `l = ['1', 'Yash', 'Raj']`

`f.writelines(l)`

↳ readline() :- reads single line

↳ readlines() :- read multiple lines

↳ truncate(size) :- resize the file to a specific size

↳ tell() :- returns the current file location

↳ fileno() :- returns a no. that represent string from OS perspective

↳ flush() :- flushes the buffer

↳ seek() :- Places the file object at the begining of the file

⇒ Modes of opening a file

r = read w = write a = append

t = text b = binary

t = used to open in a combination of two ex:- rtw

Aus:- To perform read and write operation in a text file

⇒ Pickle - To save any object along with data python provide a module called Pickle. It is used for serializing and deserializing any python structure

Serialization - It is a process of transforming data or an object in memory (RAM) to a stream of bytes called byte stream. This byte stream in binary file can then be stored on disk or database or sent through network serialization process is also called pickling

Deserialization - It is the inverse of pickling process where the byte stream is converted back to python object

* The pickle module deals with binary file here data are not written but dump and data are not read but loaded

⇒ dump() :- Used to perform serialization
ex: dump ('data object', file object)

⇒ doing serialization :-

```
import pickle
list1 = ['1', 'Pawan', 'Bhadoriya']
f = open("Demo.dat", 'wb')
pickle.dump(list1, f)
f.close()
```

⇒ load() :- Used to perform deserialization

= Regular Expression - A regular expression or regexes is a special sequence of characters that defines a pattern for complex string matching functionality

There is module for regular expression that is "re"

⇒ search (<regex>, <string>) - it is method present in re module. It scans a string for regex match, if a match found then re.search returns a match object otherwise it returns none.

Program to find digit from string

```
import re
str = 'Pawan219singh'
m = re.search('219', str)
print(m)
```

Date: / / Page no:

Output:- SRe . SRE Match Object
span = (5, 8) Match = '19