**Ex.No:1**

**Date:                      STRUCTURED PROGRAMMING**

**AIM:**

To implement the structured programming paradigm using python.

               1a .Write a program to develop a payroll calculator for school using python. The user input starts by asking for your name or "0" to quit the program, hours worked and pay rate. Extra salary will be provided for over time according to the decision of the admin.

**ALGORITHM:**

* Start the program
* Get the input from the user
* Find the regular pay if there is no overtime hours
* Find the overtime hour.
* Find the regular time pay, over time pay and  net pay
* Print the results

**PROGRAM:**

user = str

end = "0"

hours = round(40,2)

print("One Stop Shop Payroll Calculator")

hours = (float(input("Please enter hours worked: ", )))

payrate =(float(input("Please enter your payrate: $", )))

if hours < 40:

print("Employee's name: ", user)

print("Overtime hours: 0")

print("Overtime Pay: $0.00")

regularpay = round(hours \* payrate, 2)

print("Gross Pay: $", regularpay)

elif hours > 40:

overtimehours = round(hours - 40.00,2)

print("Overtime hours: ", overtimehours)

print("Employee's name: ", user)

regularpay = round(hours \* payrate,2)

overtimerate = round(payrate \* 1.5, 2)

overtimepay = round(overtimehours \* overtimerate)

grosspay = round(regularpay+overtimepay,2)

print("Regular Pay: $", regularpay)

print("Overtime Pay: $",overtimepay)

print("Gross Pay: $", grosspay)

while user != end:

print()

user = input("Please enter your name or type '0' to quit: ")

if user == end:

print("End of Report")

**OUTPUT:**

One Stop Shop Payroll Calculator

Please enter hours worked: 5

Please enter your payrate: $35

Employee's name: <class 'str'>

Overtime hours: 0

Overtime Pay: $0.00

Gross Pay: $ 175.0

1b. Write a program to implement Compound Interest

**ALGORITHM :**

* Start the program
* Get the input principle, rate, time  from the user
* Calculate the compound interest
* Print the compound interest
* Stop

**PROGRAM :**

# Program to Calculate compound interest

principle=1000

rate=10.25

time=5

Amount = principle \* (pow((1 + rate / 100), time))

CI = Amount - principle

print("Compound interest is", CI)

**OUTPUT :**

Compound interest is 628.8946267774415

1c. Write a program to reverse a given integer

**ALGORITHM :**

* Start the program
* Assign the input value to the num variable
* Find the reverse number for the given number using while loop
* Print the reversed number
* Stop

**PROGRAM :**

num = 76542

reverse\_number = 0

print("Given Number ", num)

while num > 0:

reminder = num % 10

reverse\_number = (reverse\_number \* 10) + reminder

num = num // 10

print("Revered Number ", reverse\_number)

**OUTPUT :**

The reversed Number 24567

1d. Write a program to display the cube of the number up to a given integer

**ALGORITHM :**

* Start the program
* Assign the input value to the input\_number variable
* Find the the cube of a given number number using for loop
* Print the cube
* Stop

**PROGRAM :**

input\_number = 6

for i in range(1, input\_number + 1):

print("Current Number is :", i, " and the cube is", (i \* i \* i))

**OUTPUT :**

Current Number is : 1  and the cube is 1

Current Number is : 2  and the cube is 8

Current Number is : 3  and the cube is 27

Current Number is : 4  and the cube is 64

Current Number is : 5  and the cube is 125

Current Number is : 6  and the cube is 216

1e.Write a program to find the sum of the series 2 +22 + 222 + 2222 + .. n terms

**ALGORITHM :**

* Start the program
* Initialize the values for number of terms and start
* Find the sum of series using for loop
* Print the sum of series
* Stop

**PROGRAM :**

number\_of\_terms = 5

start = 2

sum = 0

for i in range(0, number\_of\_terms):

print(start, end=" ")

sum += start

start = (start \* 10) + 2

print("\nSum of above series is:", sum)

**OUTPUT :**

**2 22 222 2222 22222**

Sum of the above series: 24690

1f.Write a program to add two matrices using nested loop

**ALGORITHM:**

* Start the program
* Initialize the X,Y and result matrices
* Iterate through rows and columns
* Print the result using for loop
* Stop

**PROGRAM:**

X = [[12,7,3],

        [4 ,5,6],

       [7 ,8,9]]

Y = [[5,8,1],

        [6,7,3],

       [4,5,9]]

result = [[0,0,0],

              [0,0,0],

              [0,0,0]]

# iterate through rows

for i in range(len(X)):

   # iterate through columns

   for j in range(len(X[0])):

       result[i][j] = X[i][j] + Y[i][j]

for r in result:

   print(r)

**OUTPUT :**

[17, 15, 4]

[10, 12, 9]

[11, 13, 18]

1g.Write a program to convert kilometer to miles

**ALGORITHM :**

* Start the program
* Get the input  from the user
* Initialize the conversion factor
* Convert to miles
* Stop

**PROGRAM :**

# Taking kilometers input from the user

kilometers = float(input("Enter value in kilometers: "))

conv\_fac = 0.621371

# calculate miles

miles = kilometers \* conv\_fac

print('%0.2f kilometers is equal to %0.2f miles' %(kilometers,miles))

**OUTPUT :**

Enter value in kilometers: 3.5

3.50 kilometers is equal

**RESULT:**

Thus the Python  program to implement payroll calculator, compound interest, to reverse a given integer, to display a cube of a given number, to find the sum of series, to add two matrices and to convert from kilometer to miles  are executed successfully.

**Ex.No:2**

**Date:**

**PROCEDURAL PROGRAMMING**

**AIM:**

To write a Python Programs to implement procedural programming paradigm

2a. Write a program to generate list of random numbers using procedure

**ALGORITHM  :**

* Start the program
* Import the random package
* Get the random numbers
* Append the random numbers to the list
* Print the random numbers
* Stop

**PROGRAM  :**

import random

def Rand(start, end, num):

res = []

for j in range(num):

res.append(random.randint(start, end))

return res

# Driver Code

num = 10

start = 20

end = 40

print(type(Rand))

print(Rand(start, end, num))

**OUTPUT  :**

<class 'function'>

[26, 36, 26, 35, 20, 33, 29, 25, 37, 33]

2b.Write a program to return the largest Fibonacci number smaller than x and the lowest fibonacci number higher than x.

**ALGORITHM  :**

* Start the program
* Get the integer from user
* Find largest Fibonacci Number smaller than x
* Find smallest Fibonacci Number larger than x:
* print largest Fibonacci Number smaller than x
* print smallest Fibonacci Number larger than x
* Stop

**PROGRAM  :**

def fib\_intervall(x):

if x < 0:

return -1

(old,new) = (0,1)

while True:

if new < x:

(old,new) = (new,old+new)

else:

if new == x:

new = old+new

return (old, new)

while True:

x = int(input("Your number: "))

if x <= 0:

break

(lub, sup) = fib\_intervall(x)

print("Largest Fibonacci Number smaller than x: " + str(lub))

print("Smallest Fibonacci Number larger than x: " + str(sup))

**OUTPUT :**

Your number: 5

Largest Fibonacci Number smaller than x: 3

Smallest Fibonacci Number larger than x: 8

Your number: 4

Largest Fibonacci Number smaller than x: 3

Smallest Fibonacci Number larger than x: 5

Your number: 9

Largest Fibonacci Number smaller than x: 8

Smallest Fibonacci Number larger than x: 13

Your number: -1

2c. Write a Python program to make a chain of function decorators (bold, italic, underline etc.).

**ALGORITHM:**

* Define the make\_bold function and wrapped function that is defined inside make\_bold to create bold letters .
* Define the make\_italic function and wrapped function that is defined inside make\_italic to create italic letters .
* Define the make\_underline function and wrapped function that is defined inside make\_underline to create bold letters .
* Call make\_bold, make\_italic and make\_underline functions

**PROGRAM:**

def make\_bold(fn):

    def wrapped():

        return "<b>" + fn() + "</b>"

    return wrapped

def make\_italic(fn):

    def wrapped():

        return "<i>" + fn() + "</i>"

    return wrapped

def make\_underline(fn):

    def wrapped():

        return "<u>" + fn() + "</u>"

    return wrapped

@make\_bold

@make\_italic

@make\_underline

def hello():

    return "hello world"

print(hello()) ## returns "<b><i><u>hello world</u></i></b>"

**OUTPUT:**

"<b><i><u>hello world</u></i></b>"

2d.Write a Python program to access a function inside a function.

**ALGORITHM:**

* Define a test function
* Define add function inside test function
* Perform addition operation
* Return the result
* Call the test function
* Print the result

**PROGRAM:**

def test(a):

        def add(b):

                nonlocal a

                a += 1

                return a+b

        return add

func= test(4)

print(func(4))

**OUTPUT:**

9

2e.Write a Python function that takes a list and returns a new list with unique elements of the first list.

**ALGORITHM:**

* Define unique\_list function
* Cheeck the uniqueness of a each value
* return the unique value
* print the unique list

**PROGRAM**:

def unique\_list(l):

x = []

for a in l:

if a not in x:

x.append(a)

return x

print(unique\_list([1,2,3,3,3,3,4,5]))

**OUTPUT:**

[1, 2, 3, 4, 5]

2f.Write a Python function to check whether a string is a pangram or not.

**ALGORITHM:**

* Import string and sys packages
* Define a function ispangram
* Check every letter of alphabet is present
* Call  the function and check with the quick brown fox jumps over the lazy dog
* Print the result

**PROGRAM:**

import string, sys

def ispangram(str1, alphabet=string.ascii\_lowercase):

alphaset = set(alphabet)

return alphaset<= set(str1.lower())

print ( ispangram('The quick brown fox jumps over the lazy dog'))

**OUTPUT:**

True

**RESULT:**

Thus the Python program to generate list of random numbers, to return the largest Fibonacci number smaller than x and the lowest fibonacci number higher than x, to create chain of function decorators, to access function inside the function, to return unique elements of list from existing list have been executed successfully.

**Ex.No:3**

**Date:**

**OBJECT ORIENTED PROGRAMMING**

**AIM:**

To Write a Python Programs to implement Object Oriented Programming Paradigm

3a. Write a program to create bank account and to perform deposit and withdraw operations using class and objects

**ALGORITHM:**

* Start the program
* Create class named Bank Account
* Initialize the constructor to make the balance zero
* Define and implement the withdraw operation.
* Define and implement the deposit operation.
* Create  the  object
* Call the withdraw and deposit function using object
* Stop

**PROGRAM:**

class BankAccount:

def \_\_init\_\_(self):

self.balance = 0

def withdraw(self, amount):

self.balance -= amount

return self.balance

def deposit(self, amount):

self.balance += amount

return self.balance

a = BankAccount()

b = BankAccount()

print(a.deposit(100))

print(b.deposit(50))

print(b.withdraw(10))

print(a.withdraw(10))

**OUTPUT :**

100

50

40

90

3b.Write a program to create employee class using constructor and destructor and  to get ID, name, gender , city and salary

**ALGORITHM:**

* Start the program
* Initialize all the values using constructor.
* Initialize the destructor
* Get the input from user.
* Display the data
* Create the object for the employee class
* Call functions using class
* Stop

**PROGRAM :**

class Employee:

def \_\_init\_\_(self): #Constructor

self.\_\_id = 0

self.\_\_name = ""

self.\_\_gender = ""

self.\_\_city = ""

self.\_\_salary = 0

print("Object Initialized.")

def \_\_del\_\_(self): #Destructor

print("Object Destroyed.")

def setData(self):

self.\_\_id=int(input("Enter Id\t:"))

self.\_\_name = input("Enter Name\t:")

self.\_\_gender = input("Enter Gender:")

self.\_\_city = input("Enter City\t:")

self.\_\_salary = int(input("Enter Salary:"))

def \_\_str\_\_(self):

data = "["+str(self.\_\_id)+","+self.\_\_name+","+self.\_\_gender+","+self.\_\_city+","+str(self.\_\_salary)+"]"

return data

def showData(self):

print("Id\t\t:",self.\_\_id)

print("Name\t:", self.\_\_name)

print("Gender\t:", self.\_\_gender)

print("City\t:", self.\_\_city)

print("Salary\t:", self.\_\_salary)

def main():

#Employee Object

emp=Employee()

emp.setData()

emp.showData()

print(emp)

if \_\_name\_\_=="\_\_main\_\_":

main()

**OUTPUT :**

Object Initialized.

Enter Id        :101

Enter Name      :Pankaj

Enter Gender:Male

Enter City      :Delhi

Enter Salary:70000

Id              : 101

Name    : Pankaj

Gender  : Male

City    : Delhi

Salary  : 70000

[101,Pankaj,Male,Delhi,70000]

Object Destroyed.

3c.To create the student class that consists of name, id and age attribute and to create the object of the student, to print attribute name of the object, to reset the value of the age, to print the modified value of age  , to print true if the student contains the attribute with name and  to delete the attribute age.

**ALGORITHM :**

* Start the program
* Create the student class with name , id and age.
* Create the object of the student class.
* Print attribute name of the object.
* Reset the value of attribute age to 23
* Prints the modified value of age
* Delete the attribute’s age.
* Stop

**PROGRAM :**

class Student:

def \_\_init\_\_(self, name, id, age):

self.name = name

self.id = id

self.age = age

# creates the object of the class Student

s = Student("John", 101, 22)

# prints the attribute name of the objects

print(getattr(s, 'name'))

# reset the value of attribute age to 23

setattr(s, "age", 23)

# prints the modified value of age

print(getattr(s, 'age'))

print(hasattr(s, 'id'))

# deletes the attribute age

delattr(s, 'age')

**OUTPUT  :**

John

23

True

AttributeError: 'Student' object has no attribute 'age'

3d.To implement the object oriented concepts.There are 258 computers available in computer programming lab where each computers are used eight hours per day. Write a Python  program using classes and objects that contain getDetail() for getting input from  user,calculatesecondperDay() for calculating the usage of each computer in seconds per day, calculateminutesperWeek() for calculating the usage of each computer in minutes per week ,calculatehourperMonth() for calculating usage of each computer in hour per month and calculatedayperYear() for calculating usage of each computer in day per yearList all the Components of structured programming language.

**ALGORITHM  :**

* Start the program
* Create the  calc class with getdetail() to get hours.
* Define calcultesecondsperday() function  to calculate seconds peer day.
* Define calculateminutesperweek() function to calculate minutes in a week.
* Create calculatedaysperyear() function to calculate no. of days in a year.
* Define calculatehourspermonth() function  to compute hours per month.
* Define an object and call the functions
* Stop

**PROGRAM :**

class calc:

def getDetail(self):

self.total\_computer=258

self.total\_hour=6

def calculatesecondsperDay(self):

Second\_per\_Day=self.total\_hour\*60\*60

print('Total Seconds per Day:',Second\_per\_Day)

def calculateminutesperWeek(self):

Minutes\_per\_Week=self.total\_hour\*60\*7

print("Total Minutes per Week:",Minutes\_per\_Week)

def calculatehourperMonth(self):

Hour\_per\_Month=self.total\_hour\*30

print("Total Hour per Month:",Hour\_per\_Month)

def calculatedayperyear(self):

Day\_per\_Year=(self.total\_hour\*365)/24

print("Total Day per Year:",Day\_per\_Year)

to=calc()

to.getDetail()

to.calculatesecondsperDay()

to.calculateminutesperWeek()

to.calculatehourperMonth()

to.calculatedayperyear()

**OUTPUT  :**

Total Seconds per Day: 28800

Total Minutes per Week: 3360

Total Hour per Month: 240

Total Day per Year: 121.66666666666667