Index

1.WAP to demonstrate overloading class

```python
# parent class
class Person:
  #costructor made of two varriables
  def __init__(self, fname, lname):
    self.firstname = fname
    self.lastname = lname
  def printname(self):
    print(self.firstname, self.lastname)
#child class Student drived from Person
class Student(Person):
  def __init__(self, fname, lname, year):
    super().__init__(fname, lname)
    self.graduationyear = year
  def welcome(self):
    print("Welcome", self.firstname, self.lastname, "to the class of", self.graduationyear)

x = Student("Mike", "Olsen",2019)
x.printname() #this function is drived from its base class
x.welcome()
```

Output:

```
PS C:\Users\Dell\Desktop\MSC CS\SEM-2\python_and_c\python prg> python -u "c:\
Users\Dell\Desktop\MSC CS\SEM-2\python_and_c\python prg\overload_class.py"
preeti Bhosale
Welcome preeti Bhosale to the class of 2022
PS C:\Users\Dell\Desktop\MSC CS\SEM-2\python_and_c\python prg>
```

2.WAP to find factorial using return function

```python
# funtion return withoght argument
def find_factorial():
    num=int(input("Enter a number to find factorial :"))
    fact=1
    for i in range(1,num+1):
        fact=fact*i
    return (fact)
f1=find_factorial()
print("The factorial is :",f1)
```

Output:

```
PS C:\Users\Dell\Desktop\MSC CS\SEM-2\python_and_c\python prg> python -u "c:\
Users\Dell\Desktop\MSC CS\SEM-2\python_and_c\python prg\facto.py"
Enter a number to find factorial :4
The factorial is : 24
PS C:\Users\Dell\Desktop\MSC CS\SEM-2\python_and_c\python prg>
```

3.WAP to find prime or not (user input)
```python
num=int(input("Enter any Number :"))
flag=0
if (num==1):
    print(num," is not prime number")
elif (num>=2):
    for i in range(2,num):
        if (num%i==0):
            flag=1
            break
    if (flag==0):
        print("This is prime number")
    else:
        print("This is not prime number")
```

Output:

```
PS C:\Users\Dell\Desktop\MSC CS\SEM-2\python_and_c\python prg> python -u "c:\
Users\Dell\Desktop\MSC CS\SEM-2\python_and_c\python prg\prime.py"
Enter any Number :57
This is not prime number
PS C:\Users\Dell\Desktop\MSC CS\SEM-2\python_and_c\python prg>
```

4.WAP to find prime numbers between 100

```python
for num in range(2,100):
    flag=0
    if (num==1):
        print(num," is not prime number")
    elif (num>=2):
        for i in range(2,num):
            if (num%i==0):
                flag=1
                break
        if (flag==0):
            print(num, end="\t")
```

Output:

```
PS C:\Users\Dell\Desktop\MSC CS\SEM-2\python_and_c\python prg> python -u "c:\
Users\Dell\Desktop\MSC CS\SEM-2\python_and_c\python prg\prime100.py"
2       3       5       7       11      13      17      19      23      29  3
1       37      41      43      47      53      59      61      67      71  7
3       79      83      89      97
PS C:\Users\Dell\Desktop\MSC CS\SEM-2\python_and_c\python prg>
```

5.WAP to types of variables

```python
# integer variable.
a=100
print("The type of variable having value", a, " is ", type(a))
# float variable.
b=20.345
print("The type of variable having value", b, " is ", type(b))
# complex variable.
c=10+3j
print("The type of variable having value", c, " is ", type(c))
#bool
d=True
print("The type of variable having value", d, " is ", type(d))
e= "Hello World"
print("The type of variable having value", e, " is ", type(e))
f=None
print("The type of variable having value", f, " is ", type(f))
```

Output:

```
Users\Dell\Desktop\MSC CS\SEM-2\python_and_c\python prg\code.py"
The type of variable having value 100  is  <class 'int'>
The type of variable having value 20.345  is  <class 'float'>
The type of variable having value (10+3j)  is  <class 'complex'>
The type of variable having value True  is  <class 'bool'>
The type of variable having value Hello World  is  <class 'str'>
The type of variable having value None  is  <class 'NoneType'>
PS C:\Users\Dell\Desktop\MSC CS\SEM-2\python_and_c\python prg>
```

6.WAP program to demonstrate Arithmatic Operations
```
a = 21
b = 10
# Addition
print ("a + b : ", a + b)
# Subtraction
print ("a - b : ", a - b)
# Multiplication
print ("a * b : ", a * b)

# Division
print ("a / b : ", a / b)
# Modulus
print ("a % b : ", a % b)
# Exponent
print ("a ** b : ", a ** b)
# Floor Division
print ("a // b : ", a // b)
```

Output:

```
PS C:\Users\Dell\Desktop\MSC CS\SEM-2\python_and_c\python prg> python -u "c:\
Users\Dell\Desktop\MSC CS\SEM-2\python_and_c\python prg\code.py"
a + b :  31
a - b :  11
a * b :  210
a / b :  2.1
a % b :  1
a ** b :  16679880978201
a // b :  2
PS C:\Users\Dell\Desktop\MSC CS\SEM-2\python_and_c\python prg>
```

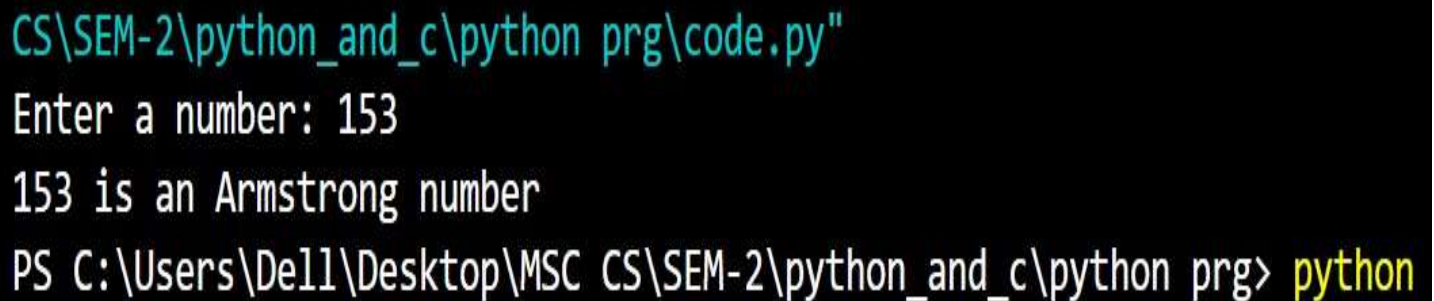7. WAP to check if the number is an Armstrong number or not

```
# take input from the user
num = int(input("Enter a number: "))

# initialize sum
sum = 0

# find the sum of the cube of each digit
temp = num
while temp > 0:
    digit = temp % 10
    sum += digit ** 3
    temp //= 10

# display the result
if num == sum:
    print(num,"is an Armstrong number")
else:
    print(num,"is not an Armstrong number")
```

Output:

```
CS\SEM-2\python_and_c\python prg\code.py"
Enter a number: 153
153 is an Armstrong number
PS C:\Users\Dell\Desktop\MSC CS\SEM-2\python_and_c\python prg> python
```

8.WAP to demonstrate String Methods

```python
str1 = " AbcDEfghIJ "
print(str1.upper())
print(str1.lower())
print(str1.strip)
str2 = "Silver Spoon"
print(str2.replace("Sp", "M"))
str2 = "Silver Spoon"
print(str2.split(" "))     #Splits the string at the whitespace " ".
str1 = "hello"
capStr1 = str1.capitalize()
print(capStr1)
str2 = "hello WorlD"
capStr2 = str2.capitalize()
print(capStr2)
str1 = "Welcome to the Console!!!"
print(str1.center(50))
str2 = "Abracadabra"
countStr = str2.count("a")
print(countStr)
```

Output:
ABCDEFGHIJ
abcdefghij
Silver Spoon
Silver Moon
['Silver', 'Spoon']
Hello
Hello world
        Welcome to the Console!!!
4

9.WAP to demonstrate list methods.

```python
Colors1 = ["violet", "indigo", "blue", "green"]
Colors1.sort()
print(Colors1)
colors.sort(reverse=True)
print(colors)
colors = ["violet", "green", "indigo", "blue", "green"]
print(colors.index("green"))
print(colors.count("green"))
newlist = colors.copy()
print(colors)
print(newlist)
colors.append("purple")
print(colors)
colors.extend(Colors1)
print(colors)
```

10. WAP on decision making example

```python
country = ("Spain", "Italy", "India", "England", "Germany")
if "Russia" in country:
    print("Russia is present.")
else:
    print("Russia is absent.")
```

```
Users\Dell\Desktop\MSC CS\SEM-2\python_and_c\python prg\code.py"
['blue', 'green', 'indigo', 'violet']
1
2
['violet', 'green', 'indigo', 'blue', 'green']
['violet', 'green', 'indigo', 'blue', 'green']
['violet', 'green', 'indigo', 'blue', 'green', 'purple']
['violet', 'green', 'indigo', 'blue', 'green', 'purple', 'blue', 'green', 'in
digo', 'violet']
PS C:\Users\Dell\Desktop\MSC CS\SEM-2\python_and_c\python prg> []
```

11.WAP to raise zeroDivision Exception with sutaible example.

```python
def divide_numbers(x, y):
    try:
        result = x / y
        print("Result:", result)
    except ZeroDivisionError:
        print("The division by zero operation is not allowed.")
# Usage
numerator = 100
denominator =int(input("Enter denominator:"))
divide_numbers(numerator, denominator)
```
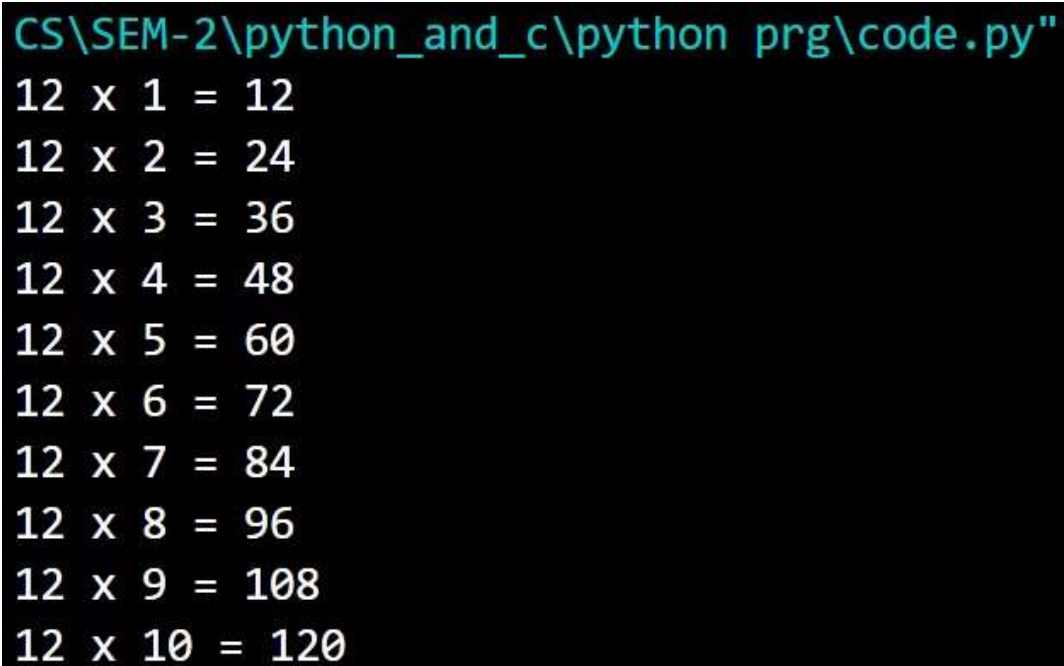
Outpur:

```
CS\SEM-2\python_and_c\python prg\code.py"
Enter denominator:0
The division by zero operation is not allowed.
PS C:\Users\Dell\Desktop\MSC CS\SEM-2\python_and_c\python prg>
```

12. WAP to create Multiplication table (user input number)

num = 12

To take input from the user

Iterate 10 times from i = 1 to 10

for i in range(1, 11):

  print(num, 'x', i, '=', num*i)

Output:

```
CS\SEM-2\python_and_c\python prg\code.py"
12 x 1 = 12
12 x 2 = 24
12 x 3 = 36
12 x 4 = 48
12 x 5 = 60
12 x 6 = 72
12 x 7 = 84
12 x 8 = 96
12 x 9 = 108
12 x 10 = 120
```

13.WAP to demonstrate multilevel inheritance
```python
class Animal:
    def speak(self):
        print("Animal Speaking")
#The child class Dog inherits the base class Animal
class Dog(Animal):
    def bark(self):
        print("dog barking")
#The child class Dogchild inherits another child class Dog
class DogChild(Dog):
    def eat(self):
        print("Eating bread...")
d = DogChild()
d.bark()
d.speak()
d.eat()
```

Output:

```
CS\SEM-2\python_and_c\python prg\code.py"
dog barking
Animal Speaking
Eating bread...
PS C:\Users\Dell\Desktop\MSC CS\SEM-2\python_and_c\python prg>
```
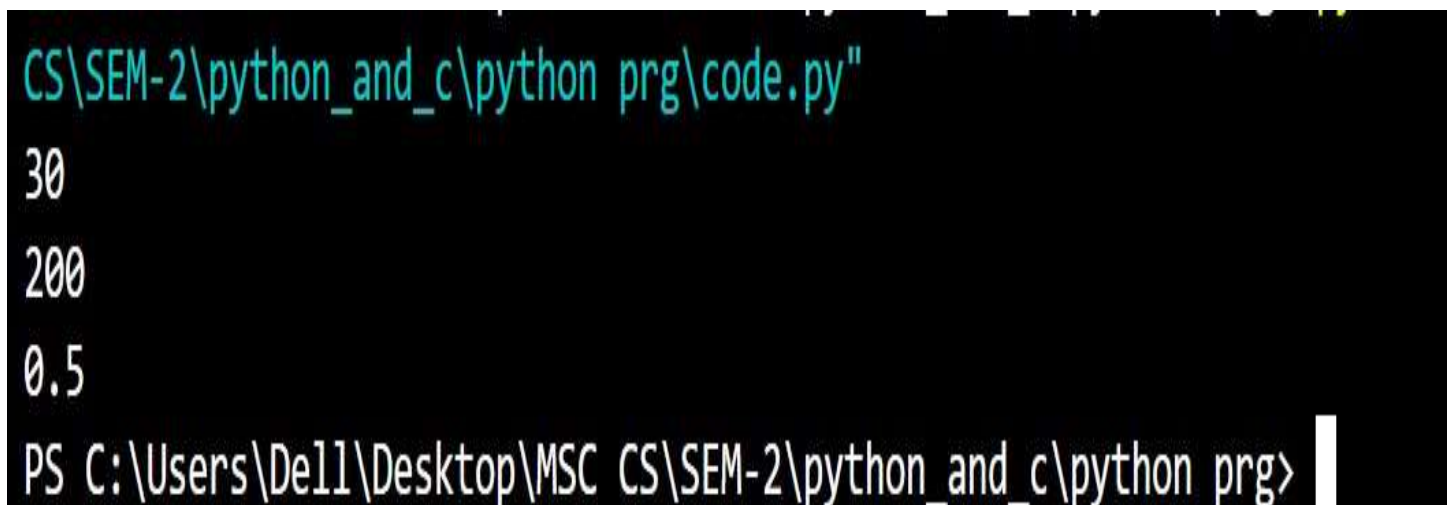
14. WAP to demonstrate multiple inheritance
```python
class Calculation1:
    def Summation(self,a,b):
        return a+b;
class Calculation2:
    def Multiplication(self,a,b):
        return a*b;
class Derived(Calculation1,Calculation2):
    def Divide(self,a,b):
        return a/b;
d = Derived()
print(d.Summation(10,20))
print(d.Multiplication(10,20))
print(d.Divide(10,20))
```

Output:


```
CS\SEM-2\python_and_c\python prg\code.py"
30
200
0.5
PS C:\Users\Dell\Desktop\MSC CS\SEM-2\python_and_c\python prg>
```

15. WAP to demonstrate hierarchical inheritance

```python
class Brands:                    #parent_class
    brand_name_1 = "Amazon"
    brand_name_2 = "Ebay"
    brand_name_3 = "OLX"


class Products(Brands):          #child_class
    prod_1 = "Online Ecommerce Store"
    prod_2 = "Online Store"
    prod_3 = "Online Buy Sell Store"


class Popularity(Brands):        #grand_child_class
    prod_1_popularity = 100
    prod_2_popularity = 70
    prod_3_popularity = 60


class Value(Brands):
    prod_1_value = "Excellent Value"
    prod_2_value = "Better Value"
    prod_3_value = "Good Value"
obj_1 = Products()          #Object_creation
obj_2 = Popularity()
obj_3 = Value()
print(obj_1.brand_name_1+" is an "+obj_1.prod_1)
print(obj_1.brand_name_1+" is an "+obj_1.prod_1)
print(obj_1.brand_name_1+" is an "+obj_1.prod_1)
```

Output:

```
PS C:\Users\Dell\Desktop\MSC CS\SEM-2\python_and_c\python prg> python -u "c:\Users\Dell\Desktop\MSC
CS\SEM-2\python_and_c\python prg\code.py"
Amazon is an Online Ecommerce Store
Amazon is an Online Ecommerce Store
Amazon is an Online Ecommerce Store
PS C:\Users\Dell\Desktop\MSC CS\SEM-2\python_and_c\python prg>
```

16. WAP to demonstrate Hybrid_inheritance

```python
class PC:
def fun1(self):
print("This is PC class")

class Laptop(PC):
def fun2(self):
print("This is Laptop class inheriting PC class")

class Mouse(Laptop):
def fun3(self):
print("This is Mouse class inheriting Laptop class")

class Student(Mouse, Laptop):
def fun4(self):
print("This is Student class inheriting PC and Laptop")

# Driver's code
obj = Student()
obj1 = Mouse()
obj.fun4()
obj.fun3()
```

Output:

17.WAP to demonstrate polymorphism

```python
class Vehicle:
  def __init__(self, brand, model):
    self.brand = brand
    self.model = model
  def move(self):
    print("Move!")


class Car(Vehicle):
  pass
class Boat(Vehicle):
  def move(self):
    print("Sail!")
class Plane(Vehicle):
  def move(self):
    print("Fly!")
car1 = Car("Ford", "Mustang") #Create a Car object
boat1 = Boat("Ibiza", "Touring 20") #Create a Boat object
plane1 = Plane("Boeing", "747") #Create a Plane object
for x in (car1, boat1, plane1):
  print(x.brand)
  print(x.model)
  x.move()
```

Output:

```
"c:/Users/Dell/Desktop/MSC CS/SEM-2/python_and_c/python prg/code.py"
Ford
Mustang
Move!
Ibiza
Touring 20
Sail!
Boeing
747
Fly!
PS C:\Users\Dell\Desktop\MSC CS\SEM-2\python_and_c\python prg>
```

18. WAP to Demonstrate operator overloading

```python
class complex_1:
    def __init__(self, X, Y):
        self.X = X
        self.Y = Y
 # Now, we will add the two objects
    def __add__(self, U):
        return self.X + U.X, self.Y + U.Y


Object_1 = complex_1(23, 12)
Object_2 = complex_1(21, 22)
Object_3 = Object_1 + Object_2
print (Object_3)
```

Output:
(44, 34)

19.WAP to swap value using two variable

```
a=int(input("Enter any number for a:"))
b=int(input("Enter any number for b:"))
print("A: ",a)
print("B: ",b)
a=a+b
b=a-b
a=a-b
print("after swap")
print("A: ",a)
print("B: ",b)
```
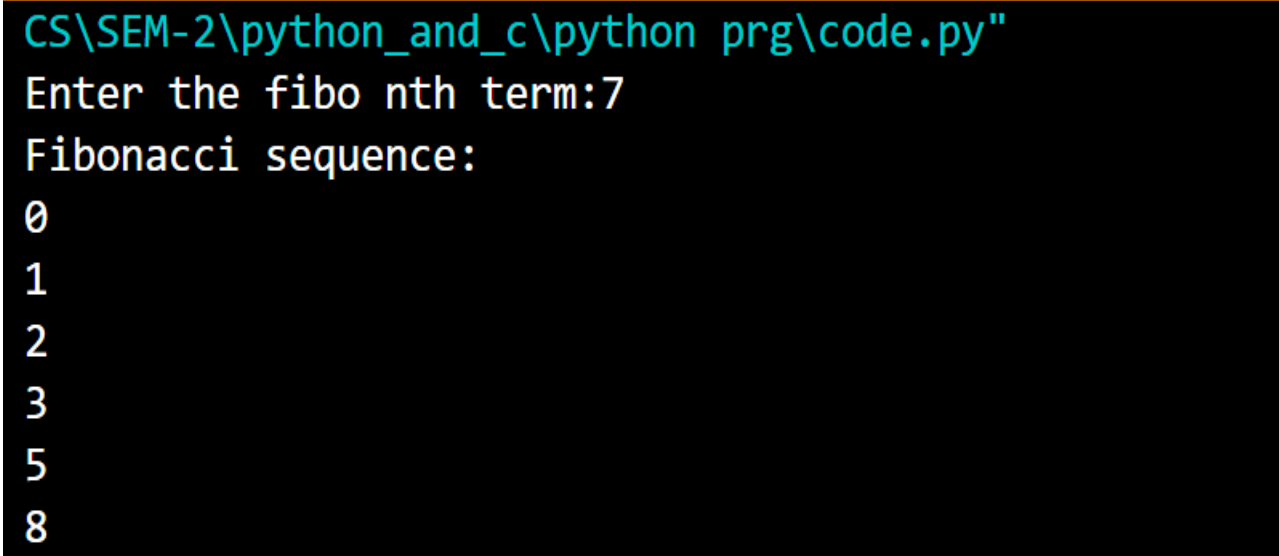
Output:

```
PS C:\Users\Dell\Desktop\MSC CS\SEM-2\python_and_c\python prg> & C:/U
"c:/Users/Dell/Desktop/MSC CS/SEM-2/python_and_c/python prg/swap.py"
Enter any number for a:13
Enter any number for b:18
A:  13
B:  18
after swap
A:  18
B:  13
PS C:\Users\Dell\Desktop\MSC CS\SEM-2\python_and_c\python prg> []
```

20. Write a program to display the Fibonacci sequence(using recursion)

```python
def recur_fibo(n):
    if n <= 1:
        return n
    else:
        return(recur_fibo(n-1) + recur_fibo(n-2))

nterms = int(input("Enter the fibo nth term:")
# check if the number of terms is valid
if nterms <= 0:
    print("Plese enter a positive integer")
else:
    print("Fibonacci sequence:")
    for i in range(nterms):
        print(recur_fibo(i))
```

Output:

```
CS\SEM-2\python_and_c\python prg\code.py"
Enter the fibo nth term:7
Fibonacci sequence:
0
1
2
3
5
8
```

21.WAP to Remove extra chater from the string
# define punctuation
punctuations = '''!()-[]{};:'"\,<>./?@#$%^&*_~'''

my_str = "Hello!!!, he said ---and went."

To take input from the user
my_str = input("Enter a string: ")

# remove punctuation from the string
no_punct = ""
for char in my_str:
    if char not in punctuations:
        no_punct = no_punct + char

# display the unpunctuated string
print(no_punct)

Output:

```
CS\SEM-2\python_and_c\python prg\code.py"
Before string: Hello!!!, he said ---and went.
Hello he said and went
PS C:\Users\Dell\Desktop\MSC CS\SEM-2\python_and_c\python prg> 
```