**Data Structures**

**Practice Assignment (Python Programming)**

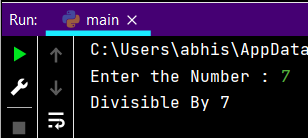
S Abhishek

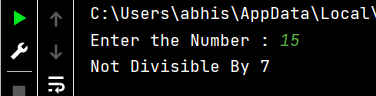
AM.EN.U4CSE19147

**Section A: Python Basics and Loops**

**1. Write a python program to check whether a number is divisible by 7 or not.**

n = int(input("Enter the Number : "))  
  
if n % **7** == **0**:  
 print("Divisible By 7")  
else:  
 print("Not Divisible By 7")

****

****

**Write a python program to input marks of five subjects Physics, Chemistry, Biology, Mathematics and Computer.**

**Calculate percentage and grade according to following:**

**Percentage >= 90% : Grade A**

**Percentage >= 80% : Grade B**

**Percentage >= 70% : Grade C**

**Percentage >= 60% : Grade D**

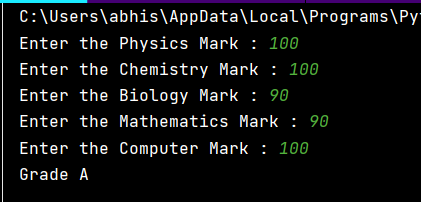
**Percentage >= 40% : Grade E**

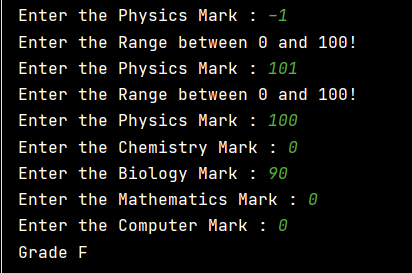
**Percentage < 40% : Grade F**

**NB:- Marks ranges from 0 – 100.**

**Need to check for the invalid inputs.**

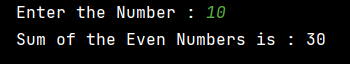
phy = -**1**che = -**1**bio = -**1**math = -**1**comp = -**1**tot = **0**while phy < **0** or phy > **100**:  
 phy = int(input("Enter the Physics Mark : "))  
 if phy < **0** or phy > **100**:  
 print("Enter the Range between 0 and 100!")  
  
while che < **0** or che > **100**:  
 che = int(input("Enter the Chemistry Mark : "))  
 if che < **0** or che > **100**:  
 print("Enter the Range between 0 and 100!")  
  
while bio < **0** or bio > **100**:  
 bio = int(input("Enter the Biology Mark : "))  
 if bio < **0** or bio > **100**:  
 print("Enter the Range between 0 and 100!")  
  
while math < **0** or math > **100**:  
 math = int(input("Enter the Mathematics Mark : "))  
 if math < **0** or math > **100**:  
 print("Enter the Range between 0 and 100!")  
  
while comp < **0** or comp > **100**:  
 comp = int(input("Enter the Computer Mark : "))  
 if comp < **0** or comp > **100**:  
 print("Enter the Range between 0 and 100!")  
  
tot = phy + che + bio + math + comp  
  
tot = (tot / **500**) \* **100**if tot >= **90**:  
 print("Grade A")  
  
elif **80** <= tot < **90**:  
 print("Grade B")  
  
elif **70** <= tot < **80**:  
 print("Grade C")  
  
elif **60** <= tot < **70**:  
 print("Grade D")  
  
elif **40** <= tot < **60**:  
 print("Grade E")  
  
else:  
 print("Grade F")

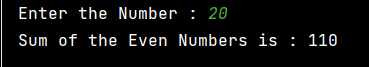
****

****

**3. Write a python program to find sum of all even numbers between 1 to n.**

n = int(input("Enter the Number : "))  
  
i = **1**sum\_of\_even = **0**while i <= n:  
 if i % **2** == **0**:  
 sum\_of\_even += i  
 i = i + **1**print("Sum of the Even Numbers is : {}".format(sum\_of\_even))

****

****

**4. Write a python program to print multiplication table of a given number.**

**NB: If the given number is 5 then the output should be in the following format**

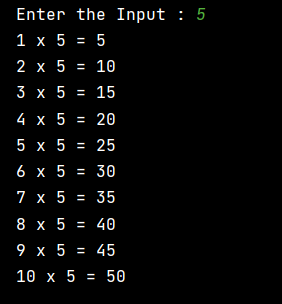
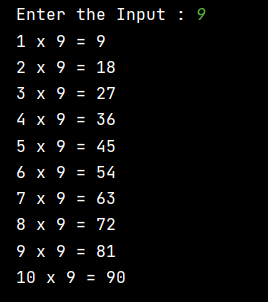
**1 x 5 = 5**

**2 x 5 = 10**

**……………..**

**10 x 5 = 50**

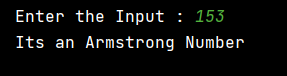
n = int(input("Enter the Input : "))  
  
for i in range(**1, 11**):  
 print("{} x {} = {}".format(i**,** n**,** i \* n))

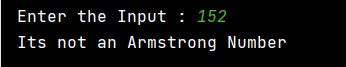
** **

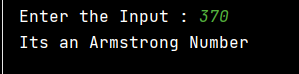
**5. Write a python program to check whether a number is Armstrong number or not.**

**Eg:- 153 is an Armstrong number. Because, 13 + 53 + 33 = 153**

n = int(input("Enter the Input : "))  
  
temp = n  
tot = **0**while temp>**0**:  
 it = temp%**10** #print(it)  
 tot = tot + (it\*it\*it)  
 temp = temp//**10**#print(tot)  
  
if tot==n:  
 print("Its an Armstrong Number")  
else:  
 print("Its not an Armstrong Number")

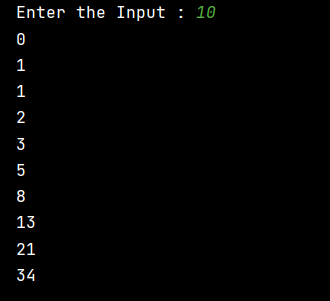
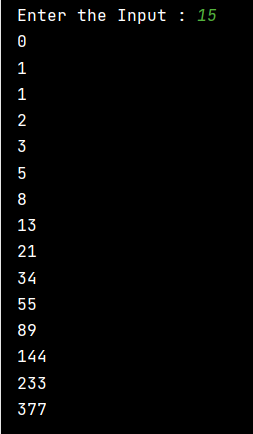
****

****

****

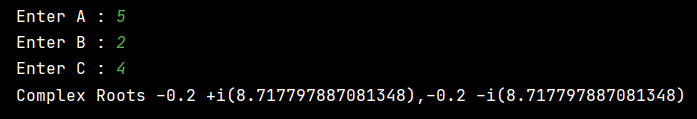
**6. Write a python program to print Fibonacci series up to n terms.**

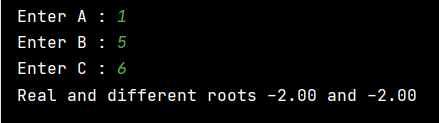
n = int(input("Enter the Input : "))  
  
a = **0**b = **1**print(a)  
  
print(b)  
  
for i in range(n-**2**):  
 c = a + b  
 print(c)  
 a = b  
 b = c

** **

**7. Write a python program to solve Quadratic Equation**

import math  
  
a = int(input("Enter A : "))  
b = int(input("Enter B : "))  
c = int(input("Enter C : "))  
  
x = **0**y = **0**determinant = (b \* b) - (**4** \* a \* c)  
  
sqrt\_val = math.sqrt(abs(determinant))  
  
if determinant == **0**:  
 print("Real and same roots {}".format(-b / (**2** \* a)))  
  
elif determinant > **0**:  
 print("Real and different roots {0:.2f} and {0:.2f}".format((-b + sqrt\_val) / (**2** \* a)**,** (-b - sqrt\_val) / (**2** \* a)))  
  
else:  
 print("Complex Roots {} +i({}),{} -i({})".format((- b / (**2** \* a))**,**sqrt\_val**,** (- b / (**2** \* a))**,**sqrt\_val))

****

****

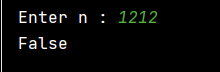
**Section B:**

**Python Function Write a Python program to create a function:**

**8. To check whether a number is palindrome or not and returns True or False respectively.**

def pal(x):  
 y = x  
 rev = **0** while y > **0**:  
 mod = y % **10** rev = (rev \* **10**) + mod  
 y = y // **10** if rev == x:  
 return True  
 else:  
 return False  
  
  
n = int(input("Enter n : "))  
print(pal(n))

****

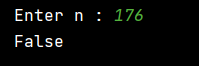
****

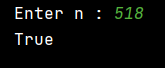
****

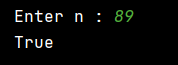
**9. That accept an integer number, find if it is Disarium number or not (Hint: A number is said to be a Disarium number when the sum of its digit raised to the power of their respective position is equal to the number itself. Eg: 175 is a Disarium number, 11 + 7 2 + 53 = 175)**

def dis(x):  
 length = len(str(x))  
  
 temp = x  
 tot = **0** while temp > **0**:  
 it = temp % **10** # print(it)  
 tot = tot + pow(it**,** length)  
 temp = temp // **10** length = length - **1** if tot == x:  
 return True  
 else:  
 return False  
  
  
n = int(input("Enter n : "))  
  
print(dis(n))

****

****

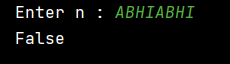
****

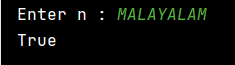
****

**10. That accepts a string, check for Palindrome and returns True or False respectively.**

def pal(x):  
 if x == x[::-**1**]:  
 return True  
 else:  
 return False  
  
  
n = (input("Enter n : "))  
  
print(pal(n))

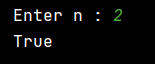
****

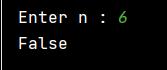
****

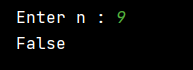
****

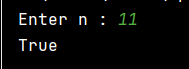
**11. To check whether a number is Prime number or not.**

def prime(x):  
 i = **2** while i <= x / **2**:  
 if x % i == **0**:  
 return False  
 i = i+**1** return True  
  
  
n = int(input("Enter n : "))  
print(prime(n))

****

****

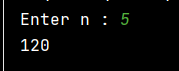
****

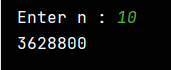
****

**Write a Python program to create a recursive function**

**12. To calculate the factorial of a number (a non-negative integer). Pass the integer as an argument and return the factorial.**

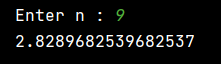
def fac(x):  
 if x == **1** or x == **0**:  
 return **1** else:  
 return x \* fac(x - **1**)  
  
  
n = int(input("Enter n : "))  
  
print(fac(n))

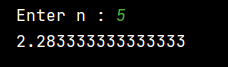
****

****

**13. That accepts a number and find the harmonic sum of that number. (Hint: the harmonic sum of n is equal to the sum of reciprocals of positive integers up to n). For example, if n =3, HS = 1+1/2+1/3**

def har(x):  
 if x == **1**:  
 return **1** else:  
 return (**1** / x) + har(x - **1**)  
  
  
n = int(input("Enter n : "))  
  
print(har(n))

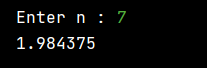
****

****

**14. That accepts a number and find the sum of geometric series of that number. (Hint: Geometric series of 4 = 1+1/2+1/4+1/8)**

def geo(x):  
 if x == **0**:  
 return **1** else:  
 return (**1** / pow(**2,** x)) + geo(x - **1**)  
  
  
n = int(input("Enter n : "))  
  
print(geo(n-**1**))



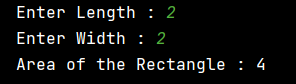


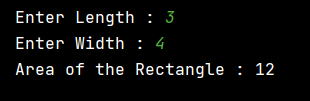
**Section C:**

**Python Object Oriented Programming**

**15. Write a Python class named Rectangle constructed by a length and width and a method which will compute the area of a rectangle.**

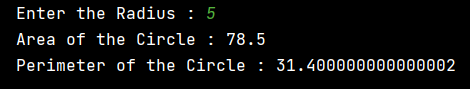
class Rectangle:  
 def \_\_init\_\_(self**,** length**,** width):  
 self.length = length  
 self.width = width  
  
 def area(self):  
 return self.length \* self.width  
  
  
l = int(input("Enter Length : "))  
w = int(input("Enter Width : "))  
  
Obj1 = Rectangle(l**,** w)  
  
print("Area of the Rectangle : {}".format(Obj1.area()))

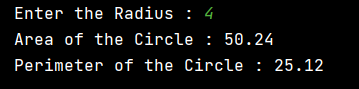
****

****

**16. Write a Python class named Circle constructed by a radius and two methods which will compute the area and the perimeter of a circle.**

class Circle:  
 def \_\_init\_\_(self**,** radius):  
 self.radius = radius  
  
 def area(self):  
 return self.radius \* self.radius \* **3.14** def perimeter(self):  
 return **2** \* **3.14** \* self.radius  
  
  
r = int(input("Enter the Radius : "))  
  
Obj1 = Circle(r)  
  
print("Area of the Circle : {}".format(Obj1.area()))  
  
print("Perimeter of the Circle : {}".format(Obj1.perimeter()))

****

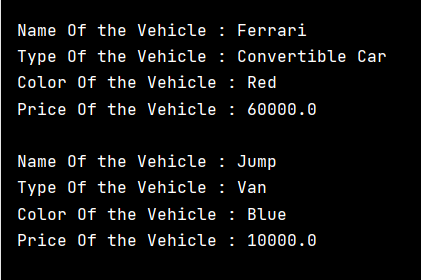
****

**17. Create a class vehicle with attributes name, type, colour, price and a method describing the details.**

**Create two new vehicles called car1 and car2. Set car1 to be a red convertible car worth $60,000.00 with a name of Ferrari, and car2 to be a blue van named Jump worth $10,000.00.**

**Your program must display the details of both cars.**

class Vehicle:  
 def \_\_init\_\_(self**,** name**,** type**,** color**,** price):  
 self.name = name  
 self.type = type  
 self.color = color  
 self.price = price  
  
 def details(self):  
 return self.name**,** self.type**,** self.color**,** self.price  
  
  
# n = input("Enter the Name : ")  
# t = input("Enter the Type : ")  
# c = input("Enter the Color : ")  
# p = input("Enter the Price : ")  
  
Car1 = Vehicle("Ferrari"**,** "Convertible Car"**,** "Red"**, 60000.00**)  
  
n**,** t**,** c**,** p = Car1.details()  
  
print("\nName Of the Vehicle : {}".format(n))  
print("Type Of the Vehicle : {}".format(t))  
print("Color Of the Vehicle : {}".format(c))  
print("Price Of the Vehicle : {}".format(p))  
  
# n = input("\nEnter the Name : ")  
# t = input("Enter the Type : ")  
# c = input("Enter the Color : ")  
# p = input("Enter the Price : ")  
  
Car2 = Vehicle("Jump"**,** "Van"**,** "Blue"**, 10000.00**)  
  
n**,** t**,** c**,** p = Car2.details()  
  
print("\nName Of the Vehicle : {}".format(n))  
print("Type Of the Vehicle : {}".format(t))  
print("Color Of the Vehicle : {}".format(c))  
print("Price Of the Vehicle : {}".format(p))

****

**18. Create a class Employee with details name, designation, department and display the details. After that, do the following:**

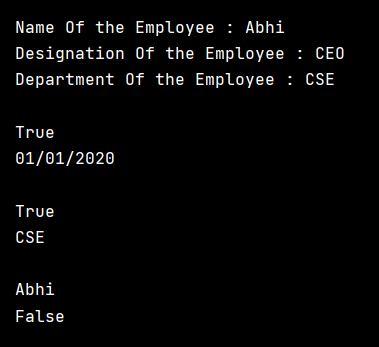
**a. Create a new attribute date of joining.**

**b. Check for department attribute.**

**c. Retrieve the employee name**

**d. Delete the attribute designation**

class Employee:  
 name = "Abhi"  
 des = "CEO"  
 dept = "CSE"  
  
  
print("\nName Of the Employee : {}".format(Employee.name))  
print("Designation Of the Employee : {}".format(Employee.des))  
print("Department Of the Employee : {}\n".format(Employee.dept))  
  
setattr(Employee**,** "Date\_of\_Joining"**,** "01/01/2020")  
  
print(hasattr(Employee**,** 'Date\_of\_Joining'))  
print("{}\n".format(getattr(Employee**,** 'Date\_of\_Joining')))  
  
print(hasattr(Employee**,** 'dept'))  
print("{}\n".format(getattr(Employee**,** 'dept')))  
  
print(getattr(Employee**,** 'name'))  
  
delattr(Employee**,** 'des')  
print(hasattr(Employee**,** 'des'))

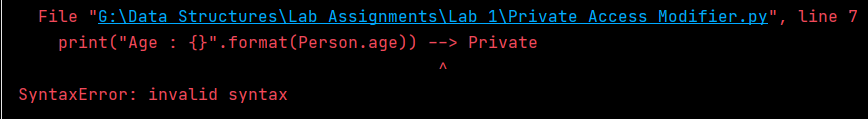
****

**19. Create a class Person with name and age as attributes.**

**Age is a private, attribute of Person which should not be permitted to access outside class.**

class Person:  
 name = "Abhi"  
 \_\_age = **19**print("Name : {}".format(Person.name))  
# print("Age : {}".format(Person.age)) --> Private

****

****

**20. Write a program to implement single inheritance.**

**a. Create the parent class Circle. Initialise the constructor with the radius of the circle.**

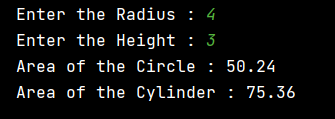
**b. Define the method get\_radius() and calc\_area() to know the radius and area of the circle.**

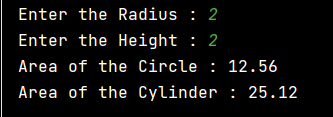
**c. Create the child class named Cylinder. Initialise the value of the height within the constructor and call the constructor of the parent class to initialise the radius of the cylinder.**

**d. Finally, defi ne the method Calc\_area() in the class Cylinder to calculate the area of the cylinder.**

**Note: Area of Cylinder = 2 \* pi \* radius \* height**

class Circle: # --> Parent Class  
 def \_\_init\_\_(self**,** radius):  
 self.radius = radius  
  
 def cal\_area(self):  
 return self.radius \* self.radius \* **3.14** def get\_area(self):  
 return self.cal\_area()  
  
  
class Cylinder(Circle): # --> Child Class  
  
 def \_\_init\_\_(self**,** radius**,** height):  
 # --> The child's \_\_init\_\_() function overrides the inheritance of the parent's \_\_init\_\_() function.  
  
 super().\_\_init\_\_(radius) # -->  
 self.height = height  
 # --> super() function makes the child class inherit all the methods and properties from its parent class.  
  
 def cal\_area(self): # --> Over rides cal\_area Method in Circle Class  
 return **2** \* **3.14** \* self.radius \* self.height  
  
 def get\_area(self): # --> Over rides get\_area Method in Circle Class  
 return self.cal\_area()  
  
  
r = int(input("Enter the Radius : "))  
h = int(input("Enter the Height : "))  
  
Obj1 = Circle(r)  
Obj2 = Cylinder(r**,** h)  
  
print("Area of the Circle : {}".format(Obj1.get\_area()))  
print("Area of the Cylinder : {}".format(Obj2.get\_area()))

****

****

**Thankyou!!**