PYTHON ASSIGNMENT

MAKE A MOVE TO PYTHON

ASSIGNMENTS

SUBMITTED TO

YASHIKA KHATRI

By: ANKUR SINGH

TASK- 7:
CLASSES AND OBJECTS

Write a program that calculates and prints the value according to the given formula:

Q= Square root of [(2*C*D)/H]

Following are the fixed values of C and H:

C is 50. H is 30.

D is the variable whose values should be input to your program in a comma-separated sequence.

```
from math import sqrt

def squrt_it(user_input):

    C, H = 50, 30
    list = []
    new_list = []
    list = user_input.split(",")

    for data in list:
        data = int(data)
        Q = sqrt((2 * C * data) / H)
        new_list.append(Q)

    print("Square root of entered values are: {}".format(new_list))


user_input = input("Please enter the numbers to get the square root: ")

squrt_it(user_input)
```

Output:

Please enter the numbers to get the square root: 10,20

Square root of entered values are: [5.773502691896258, 8.16496580927726]

Define a class named Shape and its subclass Square. The Square class has an init function which takes a length as argument. Both classes have an area function which can print the area of the shape where Shape's area is 0 by default.

```
class Shape(object):
   def __init__(self):
      pass

   def area(self):
      return 0

class Square(Shape):
   def __init__(self, length):
      Shape.__init__(self)
      self.length = length

   def area(self):
      return self.length * self.length

refernce_variable = Square(5)

print(refernce_variable.area())
```

Output:

25

Create a class to find the three elements that sum to zero from a set of n real numbers.
Input array: [-25,-10,-7,-3,2,4,8,10]
Output: [[-10,2,8],[-7,-3,10]]

```
class sum(object):
   def __init__(self,arr):
      self.arr = arr

   def find_sum(self):
      lst = self.arr
      target = 0
      result = []
      for i in range(len(lst)):
         for j in range(i+1,len(lst)):
            for k in range(j+1,len(lst)):
               a = lst[i]+lst[j]+lst[k]

               if a == 0:
                  result.append([lst[i],lst[j],lst[k]])
      return result

b = sum([-25, -10, -7, -3, 2, 4, 8, 10])
print(b.find_sum())
```

Output:

```
[[-10, 2, 8], [-7, -3, 10]]
```

What is the output of the following code? Explain your answer as well.

```
class Test:
    def __init__(self):
        self.x = 0
class Derived_Test(Test):
    def __init__(self):
        self.y = 1
def main():
    b = Derived_Test()
    print(b.x,b.y)
main()
```

Output: AttributeError: 'Derived_Test' object has no attribute 'x'
Reason: Class Derived Test inherits Class Test but variable ‘X’ isn’t inherited.

```
class A:
    def __init__(self, x= 1):
        self.x = x
class der(A):
    def __init__(self,y = 2):
        super().__init__()
        self.y = y
def main():
    obj = der()
    print(obj.x, obj.y)
main())
```

Output: In this case, we will get the output 1,2
Reason: All the classes and invoking methods are correctly defined.

```
class A:
    def __init__(self,x):
        self.x = x
    def count(self,x):
        self.x = self.x+1
class B(A):
    def __init__(self, y=0):
        A.__init__(self, 3)
        self.y = y
    def count(self):
        self.y += 1
def main():
    obj = B()
    obj.count()

    print(obj.x, obj.y)
main()
```

Output: In this case, we will get the output 3 1
Reason: All the classes and invoking methods are correctly defined.

```
class A:
    def __init__(self):
        self.multiply(15)
        print(self.i)

    def multiply(self, i):
        self.i = 4 * i;
class B(A):
    def __init__(self):
        super().__init__()

    def multiply(self, i):
        self.i = 2 * i;
```

obj = B()

Output: In this case, we will get the output 30
Reason: Because the derived class B overrides base class A

Create a Time class and initialize it with hours and minutes.
Make a method addTime which should take two time object and add them. E.g.- (2 hour and 50 min)+(1 hr and 20 min) is (4 hr and 10 min)
Make a method displayTime which should print the time.
Make a method DisplayMinute which should display the total minutes in the Time. E.g.- (1 hr 2 min) should display 62 minute

```
class  Time(object):

   def __init__(self, hours, minutes):
      self.hours = hours
      self.minutes = minutes

   def add_time(time_1, time_2):
      time_3 = Time(0, 0)
      time_3.hours = time_1.hours + time_2.hours
      time_3.minutes = time_1.minutes + time_2.minutes
      while time_3.minutes >= 60:
         time_3.hours += 1
         time_3.minutes -= 60
      return time_3

   def display_time(self):
      print("Time is {0} hours and {1} minutes".format(self.hours,
self.minutes))

   def display_minutes(self):
      print((self.hours * 60) + self.minutes, "minutes")
```

```
a = Time(2, 50)
b = Time(1, 32)
c = Time.add_time(a, b)

c.display_time()
c.display_minutes()

input()
```

Output:

```
Time is 4 hours and 22 minutes
262 minutes
```

Write a Person class with an instance variable, and a constructor that takes an integer, as a parameter. The constructor must assign to after confirming the argument passed as is not negative; if a negative argument is passed as, the constructor should set to and print Age is not valid, setting age to 0.. In addition, you must write the following instance methods:
yearPasses() should increase the instance variable by .
amIOld() should perform the following conditional actions:
If , print You are young..
If and , print You are a teenager..
Otherwise, print You are old..

Sample Input:

4

-1

10

16

18

Sample Output:

Age is not valid, setting age to 0.

You are young.

You are young.

You are young.

You are a teenager.

You are a teenager.

You are old.

You are old.

You are old.

```
class Person:

  def __init__(self,initialAge):

    if initialAge < 0:
      print("Age is not valid, setting age to 0.")
      self.initialAge = 0

    else:
      self.initialAge = initialAge

  def amIOld(self):

    if self.initialAge < 13:
      print("You are young.")

    elif self.initialAge >= 13 and self.initialAge < 18:
      print("You are a teenager.")

    else:
      print("You are old.")

  def yearPasses(self):
    self.initialAge = self.initialAge + 1

t = int(input())

for i in range(0,t):
    age = int(input())
    temp = Person(age)
    temp.amIOld()
    for j in range(0,3):
        temp.yearPasses()
    temp.amIOld()
    print("")
```

Output:

```
4
-1
Age is not valid, setting age to 0.
You are young.
You are young.

10
You are young.
You are a teenager.

16
You are a teenager.
You are old.

18
You are old.
You are old.


Process finished with exit code 0
```