CIND830 - Python Programming for Data Science

Assignment 1 (10% of the final grade)

Due on October 18, 2021 11:59 PM

This is a Jupyter Notebook document that extends a simple formatting syntax for authoring HTML and PDF. Review this website for more details on using Jupyter Notebooks.

Use the JupyterHub server on the Google Cloud Platform, provided by your designated instructor, for this assignment. Ensure using **Python 3.7.6** release then complete the assignment by inserting your Python code wherever seeing the string `#INSERT YOUR ANSWER HERE.`

When you click the `File` button, from the top navigation bar, then select `Export Notebook to HTML`, an HTML document will be generated that includes both the assignment content and the output of any embedded Python code chunks.

Use these guidelines to submit **both** the IPYNB and the exported file (HTML). Failing to submit both files will be subject to mark deduction.

Please be advised that you cannot get more than 100% in this assignment, and the **BONUS** question will only be graded if all other questions have been submitted.

Question 1 **[40 pts]**:

a) [10 pts] Write a Python program that accepts a list of marks then prints the letter grade and description according to the Grading System of the Chang School of Continuing Education.

For example, if the user enters `90, 120, -12, 71.5, 62.5`, the output would be as follows:

Mark	Letter Grade	Performance Description
90	A+	Excellent
120	Not Possible	Not Possible
-12	Not Possible	Not Possible
71.5	B-	Good
62.5	C	Satisfactory

```
input_string = input("Mark\tLetter Grade\tPerformance Description")
GradeList = []

GradeList = input_string.split(",")

for numericGrade in range(len(GradeList)):
  GradeList[numericGrade] = float(GradeList[numericGrade])

for numericGrade in GradeList:
  if numericGrade >= 90 and numericGrade <= 100:
      print(str(numericGrade) + "\tA+\t\tExcellent")
  elif numericGrade >= 85 and numericGrade < 90:
      print(str(numericGrade) + "\tA\t\tExcellent")
  elif numericGrade >= 80 and numericGrade < 85:
      print(str(numericGrade) + "\tA-\t\tExcellent")
  elif numericGrade >= 77 and numericGrade < 80:
      print(str(numericGrade) + "\B+\t\tGood")
  elif numericGrade >= 73 and numericGrade < 77:
      print(str(numericGrade) + "\tB\t\tGood")
  elif numericGrade >= 70 and numericGrade < 73:
      print(str(numericGrade) + "\tB-\t\tGood")
  elif numericGrade >= 67 and numericGrade < 70:
      print(str(numericGrade) + "\tC+\t\tSatisfactory")
  elif numericGrade >= 62 and numericGrade < 67:
      print(str(numericGrade) + "\tC\t\tSatisfactory")
  elif numericGrade >= 60 and numericGrade < 62:
      print(str(numericGrade) + "\tC-\t\tSatisfactory")
  elif numericGrade >= 57 and numericGrade < 60:
      print(str(numericGrade) + "\tD+\t\tMarginal")
```

```
  elif numericGrade >= 53 and numericGrade < 57:
      print(str(numericGrade) + "\tD\t\tMarginal")
  elif numericGrade >= 50 and numericGrade < 53:
      print(str(numericGrade) + "\tD-\t\tMarginal")
  elif numericGrade < 50 and numericGrade >= 0:
      print(str(numericGrade) + "\tF\t\tUnSatisfactory")
  else:
      print(str(numericGrade) + "\tNot Possible\tNot Possible")
```

```
Mark    Letter Grade    Performance Description90, 120, -12, 71.5, 62.5
90.0    A+              Excellent
120.0   Not Possible    Not Possible
-12.0   Not Possible    Not Possible
71.5    B-              Good
62.5    C               Satisfactory
```

link text**b) [10 pts]** Write a code to compute and print the sum of squares of only the `Excellent` marks.

For example, if the user enters `90.5, 120, -12, 71.3, 82.2, 62.5`, then the output should be:

```
The Excellent marks are: [90.5, 82.2]
The sum of squares is 14947.09
```

```
GradesList = [] #Initialize an empty list
excellentGrades = [] #Initialize an empty list

input_string = input("Mark\tSum of Squares")

GradesList = input_string.split(",")

for i in range(len(GradesList)):
 GradesList[i] = float(GradesList[i])

for item in GradesList:
  #numericGrades = [float(item) for item in input().split(",")]
```

```
    if item >= 80 and item <= 100: #excellent grades
      excellentGrades.append(item) #add to list

print("The Excellent marks are: " + str(excellentGrades))

for x in range(len(excellentGrades)):
  excellentGrades[x] = excellentGrades[x] ** 2

listsum = sum(excellentGrades)

print("The sum of squares is " + str(listsum))
```

```
Mark    Sum of Squares90.5, 120, -12, 71.3, 82.2, 62.5
The Excellent marks are: [90.5, 82.2]
The sum of squares is 14947.09
```

c) [20 pts] Write a code to compute and print the `Median`, `Average` and `Mode` of the marks between 0 and 100, inclusively, after rounding each mark to the nearest integer. Ensure notifying the user if there is no unique mode found, as the `Mode` in this exercise returns only the single most common mark in the list.

For example, if the user enters `89.7, 120, -12, 90, 60.2`, then the output should be:

```
The valid marks are: [90, 90, 60]
Median is 90 ; Average is 80 ; Mode is 90
```

However, if the user enters `90.6, 70, 100, 40.4`, then the output should be:

```
The valid marks are: [91, 70, 100, 40]
Median is 80.5 ; Average is 75.25 ; No unique mode found
```

Also, if the user enters `65, 65, 65, 35, 35, 35`, then the output should be:

```
The valid marks are: [65, 65, 65, 35, 35, 35]
```

```
Median is 50.0 ; Average is 50 ; No unique mode found
```

Hint: Use the [statistics] (https://docs.python.org/3.7/library/statistics.html) module as it provides functions to calculate mathematical statistics of numeric data.

```
#pip install statistics
validmarks = []
GradesList = []
input_string = input("Please enter the marks here, separated by a comma.")

GradesList = input_string.split(",")

for i in range(len(GradesList)):
 GradesList[i] = round(float(GradesList[i])) #round to nearest integer

for item in GradesList:
  #numericGrades = [float(item) for item in input().split(",")]
  if item >= 0 and item <= 100: #are a valid mark
    validmarks.append(item) #add to list

print("The valid marks are: " + str(validmarks))

from statistics import mode
from statistics import mean
from statistics import median
from statistics import StatisticsError

med = median(validmarks)
avg = mean(validmarks)

try:
  mod = mode(validmarks)

except StatisticsError:
  mod = "No unique mode found"

if type(mod) == type(med): #a number
  print("The Median is " + str(med) + " ; " + "Average is " + str(avg) + " ; " + "Mode is " + str(mod))
else:
```

```
print("The Median is " + str(med) + " ; " + "Average is " + str(avg) + " ; " + str(mod))
```

```
Please enter the marks here, separated by a comma.89.7, 120, -12, 90, 60.2
The valid marks are: [90, 90, 60]
The Median is 90 ; Average is 80 ; Mode is 90
```

Question 2 **[30 pts]**:

a) [10 pts] Write a code that creates a set of passwords, based on a list of enetered country names. Each password should be a combination of:

- The first three letters of the country name
- The country capital with no whitespaces
- The country currency in uppercase
- The country population in the scientific notation format

For example, if the user enters 'Canada,Egypt,India,Turkey', then the code should generate the following list:

Password
CanOttawaCAD3.55e+07
EgyCairoEGP8.77e+07
IndNewDelhiINR1.26e+09
TurAnkaraTRY7.67e+07

Hint: You can use the [countryinfo] (https://pypi.org/project/countryinfo/) library to generate the required information of the entered countries.

```
#pip install countryinfo
print("Password")
input_string = []
countrylist = []
```

```
passwordlist = []
input_string = input("Enter the names of the countries, separated by commas.")
countrylist = input_string.split(",")

from countryinfo import CountryInfo

for countryname in countrylist:
  country = CountryInfo(countryname)
  capital = country.capital()
  capital = capital.replace(" ", "") #remove whitespaces
  currency = country.currencies()

  population = country.population()
  population = "{:.2e}".format(population) #format population to two digits decimals

  password = countryname[0:3] + capital + currency[0] + population

  print(password)
  #prep for next question
  passwordlist.append(password)
```

```
Password
Enter the names of the countries, separated by commas.Canada,Egypt,India,Turkey
CanOttawaCAD3.55e+07
EgyCairoEGP8.77e+07
IndNewDelhiINR1.26e+09
TurAnkaraTRY7.67e+07
```

b) [10 pts] Count the letters of each generated password in Q2.a, then print the password along with its number of characters.

For example, if the user generated the following passwords, then the output should be as follows:

Password	Length
CanOttawaCAD3.55e+07	20
EgyCairoEGP8.77e+07	19
IndNewDelhiINR1.26e+09	22
TurAnkaraTRY7.67e+07	20

```
print("Password\t\tLength")
for item in passwordlist:
  plength = len(item)
  print(item + "\t" + str(plength))
```

```
Password                Length
CanOttawaCAD3.55e+07    20
EgyCairoEGP8.77e+07     19
IndNewDelhiINR1.26e+09  22
TurAnkaraTRY7.67e+07    20
```

c) [10 pts] Write a code to print only the passwords with more than 19 characters and include either the letter 'D', the digit '9', or the punctuation symbol '-'.

For example, the code should filter the generated passwords listed in 2.a, and print the following output:

Password
CanOttawaCAD3.55e+07
IndNewDelhiINR1.26e+09

```
print("Password")
for item in passwordlist:
  if len(item) > 19:
    if "D" in item or "9" in item or "-" in item:
      print(item)
```

```
Password
CanOttawaCAD3.55e+07
IndNewDelhiINR1.26e+09
```

BONUS [10 pts] Write a program that rounds the country population to the nearest Million

For example, if the user enters 'Canada,Egypt,India,Turkey', then the code should generate the following list:

Country	Population

Country	Population
Canada	36 Million
Egypt	88 Million
India	1264 Million
Turkey	77 Million

```
print("Country\tPopulation")
for countryname in countrylist:
  country = CountryInfo(countryname)
  population = country.population()
  #population = "{:.2e}".format(population) #format population to two digits decimals
  population = round(population, -6)
  population = str(population)
  if len(population) > 6:
    population = population[:-6] + " Million"
  print(countryname + "\t" + str(population))
```

```
   Country Population
   Canada  36 Million
   Egypt   88 Million
   India   1264 Million
   Turkey  77 Million
```

BONUS [10 pts] Design an encoder that encrypts the passwords generated in Q2.a. by replacing each character with the corresponding octal form.

For example, having the passwords listed in Q2.a., the code should display the following output:

Password	Encryption
CanOttawaCAD3.55e+07	0o1030o1410o1560o1170o1640o1640o1410o1670o1410o1030o1010o1040o630o560o650o650o1450o530o600o67
EgyCairoEGP8.77e+07	0o1050o1470o1710o1030o1410o1510o1620o1570o1050o1070o1200o700o560o670o670o1450o530o600o67
IndNewDelhiINR1.26e+09	0o1110o1560o1440o1160o1450o1670o1040o1450o1540o1500o1510o1110o1160o1220o610o560o620o660o1450o530o600o71
TurAnkaraTRY7.67e+07	0o1240o1650o1620o1010o1560o1530o1410o1620o1410o1240o1220o1310o670o560o660o670o1450o530o600o67

```
print("Password\t\tEncryption")
for item in passwordlist:
```

```
  encryptedp = ""
  for letter in item:
    letter = ord(letter) #ASCII
    encryptedp += oct(letter)#ASCII to Octal
  print(item + "\t" + str(encryptedp))
```

```
   Password                Encryption
   CanOttawaCAD3.55e+07    0o1030o1410o1560o1170o1640o1640o1410o1670o1410o1030o1010o1040o630o560o650o650o1450o530o600o67
   EgyCairoEGP8.77e+07     0o1050o1470o1710o1030o1410o1510o1620o1570o1050o1070o1200o700o560o670o670o1450o530o600o67
   IndNewDelhiINR1.26e+09  0o1110o1560o1440o1160o1450o1670o1040o1450o1540o1500o1510o1110o1160o1220o610o560o620o660o1450o53
   TurAnkaraTRY7.67e+07    0o1240o1650o1620o1010o1560o1530o1410o1620o1410o1240o1220o1310o670o560o660o670o1450o530o600o67
```

Question 3 **[30 pts]**:

Write a program that accepts the lengths of three sides of a triangle as inputs then prints the type and compute the properties of the triangle according to the listed conditions and mathematical formulae.

a) [10 pts] Ensure that all side lengths are positive and the sum of any two side lengths is greater than the third side length.

```
iLength1 = float(input("Enter the length of one of sides of the triangle.")) #side 1
iLength2 = float(input("Enter the length of the other side of the triangle.")) #side 2
iLength3 = float(input("Enter the length of the last side of the triangle.")) #side 3

if iLength1 > 0 or iLength2 > 0 or iLength3 > 0: #ensure all side lengths are positive
  if iLength1 + iLength2 > iLength3:
    print("This is ok.")
  elif iLength1 + iLength3 > iLength2:
    print("This is ok.")
  elif iLength2 + iLength3 > iLength1:
    print("This is ok.")
  else:
    print("This is not ok as the sum of any two side lengths are not greater than the third side length.")
```

```
else: #one or more of the side lengths are not positive
  print("One or more of the side lengths are not positive. Please enter positive numbers only.")
  raise SystemExit
```

```
Enter the length of one of sides of the triangle.3
Enter the length of the other side of the triangle.4
Enter the length of the last side of the triangle.5
This is ok.
```

b) [10 pts] Display the type of the triangle according to the following restrictions:

Equilateral Triangle: If the triangle has three congruent sides.

Isosceles Triangle: If the triangle has two equal sides.

Scalene Triangle: If the triangle has no congruent sides, and each side have a different length.

Right Triangle: If the square of one side equals the sum of the squares of the other two sides.

```
if iLength1 == iLength2 and iLength2 == iLength3:
  print("Equilateral")
elif iLength1 == iLength2 and iLength2 != iLength3 or iLength1 == iLength3 and iLength2 != iLength3:
  print("Isosceles")
elif iLength1 != iLength2 and iLength1 != iLength3:
  print("Scalene")

if pow(iLength1, 2) == pow(iLength2, 2) + pow(iLength3, 2) or pow(iLength2, 2) == pow(iLength1, 2) + pow(iLength3, 2) or pow
  print("Right")
```

```
Scalene
Right
```

c) [10 pts] Compute the area and the perimeter of the triangle according to the following formulae:

$$TrianglePerimeter = side_1 + side_2 + side_3$$

$$TriangleArea = \sqrt{p(p-side_1)(p-side_2)(p-side_3)}, where\ p = \frac{side_1+side_2+side_3}{2}$$

```
TrianglePerimeter = iLength1 + iLength2 + iLength3
p = TrianglePerimeter/2
TriangleArea = (p*(p-iLength1)*(p-iLength2)*(p-iLength3)) ** 0.5

print("The area of the triangle is " + str(TriangleArea) + " and the perimeter is " + str(TrianglePerimeter))
```

```
The area of the triangle is 6.0 and the perimeter is 12.0
```

This is the end of assignment 1

✓ 0s completed at 7:47 PM