CIND830F20 Assignment 1

October 17, 2020

0.1 CIND830 - Python Programming for Data Science

0.1.1 Assignment 1 (10% of the final grade)

0.1.2 Due on October 12, 2020 11:59 PM

0.1.3 Name: Tasdeed Aziz

0.1.4 ID: 500945638

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 Juputer Notebook.

Use the JupyterHub server on the Google Cloud Platform, provided by your designated instructor, for this assignment. Complete the assignment by inserting your Python code wherever you see the string "#INSERT YOUR ANSWER HERE."

When you click the `File` button, from the top navigation bar, then select `Export Notebook As ...`, a document (PDF or HTML format) will be generated that includes both the assignment content and the output of any embedded Python code chunks.

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

0.1.5 Question 1:

Based on Canada's government labour market indicators in August 2020, the unemployment population in Toronto decreased to approximately 518 thousand in August of 2020 with a month-to-month change rate of -2.5 percent.

---

a) Create four variables: `cityName` as String (Toronto), `augPopulation` as Integer (518), `changeRate` as Floating-point number (-2.5%), and `decreasing` as Boolean (True).

[49]:
```python
cityName = 'Toronto'
augPopulation = 518
changeRate = -0.025
decreasing = True
```

b) Print every literal and type of the variables in Q1.a in a separate new line.

[50]:
```python
print('\n','cityName',cityName,type(cityName),'\n','augPopulation',augPopulation,type(augPopul
 ↪'changeRate',changeRate,
      type(changeRate),'\n','decreasing',decreasing,type(decreasing))
```

```
 cityName Toronto <class 'str'>
 augPopulation 518 <class 'int'>
 changeRate -0.025 <class 'float'>
 decreasing True <class 'bool'>
```

c) If the unemployment rate stays the same in September, find how many people would still be unemployed? Round off the answer to the nearest whole number.

[51]:
```python
changePopulation = augPopulation * changeRate
sepPopulation = round(augPopulation + changePopulation)
print(sepPopulation)
```

```
505
```

d) Find the difference between the unemployment population of August and that of September.

[52]:
```python
diffUnemployment = augPopulation - sepPopulation
print(round(diffUnemployment))
```

```
13
```

e) Convert the variable `augPopulation` to a floating-point number then print the new literal and type.

[53]:
```python
augPopulation = float(augPopulation)
print('augPopulation',augPopulation, type(augPopulation))
```

```
augPopulation 518.0 <class 'float'>
```

f) Given that `var1` is the `sepPopulation` variable rounded to the nearest whole number, and `var2` is the `changeRate` variable rounded to two decimal places. Print the following statement: The unemployment populaton might be `var1` thousand in September 2020, which is a `var2` month-to-month change rate.

[54]:
```
print("The unemployment population might be",sepPopulation,"thousand in␣
↪September 2020, which is a",round(changeRate*100,2),"% month-to-month change␣
↪rate")
```

```
The unemployment population might be 505 thousand in September 2020, which is a
-2.5 % month-to-month change rate
```

0.1.6 Question 2:

Create a list of numbers called `nmbrs` starting at 0 and ending at 21, using the following code:

[55]:
```
# a list of numbers from 0 to 21
nmbrs = list(range(22))
print(nmbrs, end = ' ')
#print(type(nmbrs))
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21]
```

a) Print the `nmbrs` list in reverse order using a simple `for` loop.

[56]:
```
for i in reversed(nmbrs):
    print(i,end = ' ')
```

```
21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
```

b) Print only the numbers that are divisible by 7.

[8]:
```
for x in nmbrs:
    if x % 7 == 0:
        print(x)
```

```
0
7
14
21
```

c) Print the odd numbers that are multiple of 3.

[57]:
```
for y in nmbrs:
    if y % 2 != 0 and y % 3==0:
        print(y)
```

```
3
9
15
21
```

d) Create another list called `nmrls` that includes the corresponding english names of the numbers in the `nmbrs` list, starting at `zero` and ending at `twenty-one`. Hint: Import the inflect library, then use the `number_to_words()` function to convert numbers into words.

[58]:
```
!pip install inflect
```

```
Requirement already satisfied: inflect in /opt/conda/lib/python3.7/site-packages
(4.1.0)
Requirement already satisfied: importlib-metadata; python_version < "3.8" in
/opt/conda/lib/python3.7/site-packages (from inflect) (1.6.0)
Requirement already satisfied: zipp>=0.5 in /opt/conda/lib/python3.7/site-
packages (from importlib-metadata; python_version < "3.8"->inflect) (3.1.0)
```

[59]:
```
import inflect
a = inflect.engine()
b = a.number_to_words(nmbrs[0:10], group=1, wantlist = True)
c = a.number_to_words(nmbrs[10:22], group=2, wantlist = True)
nmrls = b + c
print(nmrls, type(nmrls))
```

```
['zero', 'one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight', 'nine',
'ten', 'eleven', 'twelve', 'thirteen', 'fourteen', 'fifteen', 'sixteen',
'seventeen', 'eighteen', 'nineteen', 'twenty', 'twenty-one'] <class 'list'>
```

e) Use a nested loop to count the letters of each element in the `nmrls` list, then print the element along with its number of letters.

[72]:
```
for char in nmrls:
    count = 0
    for letters in char:
        count = count + 1
    print(char,count)
```

```
zero 4
one 3
two 3
three 5
four 4
five 4
six 3
seven 5
eight 5
nine 4
ten 3
eleven 6
```

```
twelve 6
thirteen 8
fourteen 8
fifteen 7
sixteen 7
seventeen 9
eighteen 8
nineteen 8
twenty 6
twenty-one 10
```

f) Print the elements in the `nmrls` list that have more than or equal to 7 letters and has either the letter ‘o’ or the letter ‘g’.

[17]:
```
for character in nmrls:
    if len(character) >= 7 and 'o' in character or len(character)>=7 and 'g' in␣
 ↪character:
        print(character)
```

```
fourteen
eighteen
twenty-one
```

0.1.7 Question 3:

Write a program that accepts the lengths of three sides of a triangle as inputs, using the following code, then prints the type of the triangle according to the conditions listed below.

[21]:
```
# Acquiring Inputs
side1 = float(input("Enter the first side: "))
side2 = float(input("Enter the second side: "))
side3 = float(input("Enter the third side: "))
```

```
Enter the first side:  11
Enter the second side:  60
Enter the third side:  61
```

a) Equilateral Triangle: If the triangle has three congruent sides.

[22]:
```
if side1 == side2 == side3:
    print("Equilateral Triangle")
```

b) Isosceles Triangle: If the triangle has two equal sides.

[23]:
```
if side1 == side2 or side1 == side3 or side2 == side3 :
    print("Isosceles Triangle")
```

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

[24]:
```python
if side1 != side2 != side3:
    print("Scalene Triangle")
```

```
Scalene Triangle
```

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

[25]:
```python
if side1 ** 2 + side2 ** 2 == side3 ** 2:
    print("Right Triangle")
```

```
Right Triangle
```

e) Combine your answers in Q3a, Q3b, Q3c, and Q3d and write one program that decides the type of a given triangle according to its sides' length. Notably, a right triangle might be isosceles or scalene. (e.g. A triangle with the following sides: 3, 4, and 5 cm is a right scalene triangle)

[73]:
```python
side1 = float(input("Enter the first side: "))
side2 = float(input("Enter the second side: "))
side3 = float(input("Enter the third side: "))

if side1 == side2 == side3:
    print("Equilateral Triangle")
elif side1 == side2 or side1 == side3 or side2 == side3 :
    print("Isosceles Triangle")
else:
    print("Scalene Triangle")
if side1 ** 2 + side2 ** 2 == side3 ** 2:
    print("Right Triangle")
```

```
Enter the first side:  11
Enter the second side:  60
Enter the third side:  61

Scalene Triangle
Right Triangle
```

This is the end of assignment 1