conditons_prolongements

October 30, 2019

Sommaire

1 Structure conditionnelle avec Python

Auteur : * .laurent.fleron@gmail.com * .isabelle.detante@laposte.net
Source : A faire, au fil de l'eau ?

1.1 Informatique embarquée

1.1.1 Indicateur de charge d'une batterie de trotinette.

Difficulté (++)

Récupération d'une donnée provenant d'un capteur.

Analyse du niveau de tension de la batterie d'une trotinette électrique par un affichage d'un **barre graphe**.

Donner le nombre de kilomètres restants.

[9]:
```
# Excuter cette fonction
# coding: utf-8

def pourcentBat(tension = 12)->int:
    # Calcul du pourcentage de charge restant pour une batterie entre [11.0 et␣
 ↪14.5]
    pourcentBat = int((tension - 11.0) / (14.5 - 11.0) * 100)
```

```
    return pourcentBat

def kilo(pourcentBat):
    # Calcul du nombre de kilomètres restant
    capacite = 30

    return pourcentBat * capacite / 100
```

1.1.2 Travail demandé

Proposez une fonction "barreGraphe" demandant à l'utilisateur un float "tension" et retourne une chaine de 10 caractères comme par exemple "XXX——" correspondant à un barre graphe. Chaque X correspondant à une tranche de 10% de charge.

[16]:
```
#Ecrire votre fonction ici
# -*- coding:Utf-8 -*-

def barreGraphe ()->str:

    # Demander la tension de la batterie et enregistrer la valeur dans une␣
 ↪variable "tension"
    tension = float(input("tension de la batterie ? "))

    # Appel de la fonvtion pourcentBat
    barre = pourcentBat(tension)

    # Bloc condition pour barre inférieur à 10 %
    if barre < 10:
        print("Niveau : X--------")
        print("Mode tortue :-(")

    # Faire pour les blocs suivants
    # ...

    # Appel de la fonction kilo appelant pourcentBat et stockage dans la␣
 ↪variable capaBat
    capaBat = int(kilo(pourcentBat(tension)))

    # Affichage des kilomètres restants
    # ...

    return None
```

[17]:
```
# Testez votrefonction ici
barreGraphe()
```

```
tension de la batterie ? 13
Niveau : xxxxxx----
La batterie vous permet d'effectuer encore 17 kilomètre(s).
```

1.1.3 Pour aller plus loin…

Trouver des caracteres unicodes permettant un affichage plus évocateurs à la place des “X” et des “-”.

[11]:
```
# Une version plus simple
# -*- coding:Utf-8 -*-

def barreGraphe ()->str:

    tension = float(input("tension de la batterie ? "))

    # barre entre 0 et 10
    barre = int(pourcentBat(tension) // 10)
    # Affichage de x barre d'un caratère dont le code décimal est converti en␣
 ↪string
    # ...

    # Affichage des compléments de la barre avec un autre code
    # ...

    #  Concaténation des deux barre pleine et vide
    # ...

    # Affichage
    print (barreGraphe)

    capaBat = int(kilo(pourcentBat(tension) / 100))

    # AFfichage des kilomètres restant

    return None

barreGraphe()
```

```
  File "<ipython-input-11-fb777338cdeb>", line 15
    print(f"La batterie vous permet d'effectuer encore {capaBat}␣
↪kilomètre(s).")
                                                                  ␣
↪ ^
SyntaxError: invalid syntax
```

[12]:
```
# Pour trouver les caractères unicodes

s = ""              # chaîne vide
i = 8000             # premier code
while i <= 9620:         # dernier code
 s += chr(i)
 i = i + 1
print("Alphabet grec (minuscule) : ", s)
print (ord(""))
#
```

```
Alphabet grec (minuscule) :
```

```
9619
```

1.2 Les données structurées et leur traitement

1.2.1 Reconnaitre la langue d'un texte

On souhaite vérifier si les **occurences des voyelles** dans un texte peuvent définir la langue du texte.

Lien de textes : https://lingua.com/fr/francais/lecture/presentation/

[20]:
```
# Executer cette fonction
# coding: utf-8

def lectureTxt(fichier:str)->str:

    # Ouverture d'un fichier en lecteur seule encodé en UTF8
```

```
    with open(fichier, "r", encoding ="utf8") as f:
        # Initialisation de la vaiable totalTexte à 33
        totalTexte = ""

        # Pour toutes les lignes dans le fichier faire
        for line in f :
            # # On prend une ligne
            uneLigne = line.strip()
            # On enlève le saut de ligne
            uneLigne = uneLigne.replace('\n','')
            # Minuscule pour toutes les lettres de la ligne
            uneLigne = uneLigne.lower()
            # Ajout de la ligne au total des lignes
            totalTexte += uneLigne

    return totalTexte
```

1.2.2 Travail demandé

Difficulté (+++) *Due au formatage des valeurs de sorties*

Proposez une fonction “analyseTexte” prenant pour argument “fichier”, returnant “none” mais affichant les statistiques de texte analysé en vous inspirant du code.

[21]:
```
#Ecrire votre fonction ici
def analyseTexte (fichier:str)->str:

    # Stockage du texte dans une variable texteAnalyse
    texteAnalyse = lectureTxt(fichier)

    # Initialisation des variables
    lettreA = 0 ; lettreE = 0 ; lettreI = 0 ; lettreO = 0 ; lettreU = 0 ;␣
 ↪lettreY = 0
    compte = 0

    # Pour toutes les lignes du texte faire
    for lettre in texteAnalyse:
        # Si le texte comporte "a" compter le nombre d'occurences
        if lettre == "a":
            lettreA += 1

        # Conditions pour les autres voyelles
        # ...

        # Compte le nombre de lettres du texte
        # ...

        # Compte le nombres de voyelles
```

```
        # ...

    # Affichage des occurences des voyelles
    print (f"lettre a : {lettreA} occurence(s)")
    # ...

    # Affichage du nombre total des voyelles
    print (f"{compteVoy} voyelles pour {compte} lettres au total.\n")

    # Affichage du pourcentage de chaque voyelle dans le texte
    print("Lettre a : ", "{0:.2f}".format(round((lettreA/compte)*100,2)),"%")
    # ...

    return None
```

[22]:
```
# Testez votrefonction ici

# Choisir le fichier à analyser :

#fichier = "centreCommercial.txt"
#fichier = "famigliaGiovanni.txt"
#fichier = "ElParque.txt"
#fichier = "WonderfulFamilly.txt"
fichier = "TexteMystere.txt"

analyseTexte(fichier)
```

```
lettre a : 73 occurence(s)
lettre e : 124 occurence(s)
lettre i : 50 occurence(s)
lettre o : 26 occurence(s)
lettre u : 44 occurence(s)
lettre y : 1 occurence(s)

318 voyelles pour 975 lettres au total.

Lettre a :  7.49 %
Lettre e :  12.72 %
Lettre i :  5.13 %
Lettre o :  2.67 %
Lettre u :  4.51 %
Lettre y :  0.10 %
```

1.2.3 Pour aller plus loin...

Dans votre tableur préféré, tracer un graphe de type radar (rayons = les 6 voyelles / langue (couleurs différentes), ordonné = % de l'occurence).

Trouver pour le graphe du texte mystere la langue employée.

pour les NSI étude des K plus proches voisins pour une détection automatique. Empreinte de la langue.

1.3 Localisation

1.3.1 Savoir si un tracker gps se trouve dans une zone circulaire définie.

difficulté (+)

Comment déclencher une alerte quand un individu portant un bracelet Gps entre dans une zone dans laquelle il lui est interdit d'entrer ?

D'après la première définition du mètre (la 40 000 ème partie d'un grand cercle terrestre le rayon de la terre vaut 40000/(2*Math.PI) km soit 6366,2 km ce qui confirme sensiblement le résultat précédent, mais permet aussi de déterminer qu'un degré de grand cercle correspond à 40 000/360 soit 111,111 km puis une minute à 1852 mètres (c'est la définition du mille marin). Autrement dit , il ne reste plus qu'à déterminer l'angle des deux vecteurs OA et OB joignant le centre de la terre au deux points considérés A et B.

Le produit scalaire (XX'+YY'+ZZ') de deux vecteurs unitaires (définis par X=cos(lat)*cos(lng)*, *Y=cos(lat)*sin(lng), z=sin(lat)) portés par ces vecteurs donne le cosinus de cet angle. D'où la formule ci-dessus applicable pour calculer des distances telles que Paris - New York.

S'il s'agissait de calculer des distance des points très proches en France, mieux vaudrait alors raisonner différemment pour considérer que l'on est sur le plan de la carte de France avec des distances de 111,111 km par différence de degrés (ou 1852 mètres en minutes) sur sur l'axe Nord-Sud des latitudes (on est toujours sur un arc de grand cercle méridien) mais avec une échelle autre sur l'axe Ouest-Est des longitudes (on est alors sur un cercle de rayon réduit par le cosinus de la latitude). On considérera alors la latitude moyenne des deux points pour réduire les 111,11 km (

[25]:
```python
# Excuter cette fonction
# coding: utf-8

from math import *

def decodeTrames(trmGps1:list, trmGps2:list)->tuple:

    # ----------GPS1
    # Extraction de la chaine contenant la latitude et la longitude
    lat_a_degre = trmGps1[2]
    lon_a_degre = trmGps1[4]

    # Extraction des degrés
    lat_dda = int(trmGps1[2] / 100)
    lon_dda = int(trmGps1[4] / 100)

    # Conversion des minutes mm.mmmm en DD.DDDD
    lat_mma = (((lat_a_degre - (lat_dda * 100)) / 60))
    lon_mma = (((lon_a_degre - (lon_dda * 100)) / 60))

    # Ajout des DD + DD.DDDD
    lat_a_degre = lat_dda + lat_mma
    lon_a_degre = lon_dda + lon_mma
```

```
    # ----------GPS2
    # Conversion des DDmm.mmmmm en mm.mmmm
    lat_b_degre = trmGps2[2]
    lon_b_degre = trmGps2[4]

    # Extraction des DD
    lat_ddb = int(trmGps2[2] / 100)
    lon_ddb = int(trmGps2[4] / 100)

    # Ajout des mm.mmmm en DD.DDDD
    lat_mmb = (((lat_b_degre - (lat_ddb * 100)) / 60))
    lon_mmb = (((lon_b_degre - (lon_ddb * 100)) / 60))

    lat_b_degre = lat_ddb + lat_mmb
    lon_b_degre = lon_ddb + lon_mmb

    return (lat_a_degre, lon_a_degre, lat_b_degre, lon_b_degre)

"""
A partir de là...
https://openclassrooms.com/forum/sujet/calcul-d-une-distance-95555
"""
# Conversion des degrés en radian
def convertRad(input:float)->float:

    return (pi * input)/180

def distance(lat_a_degre:float, lon_a_degre:float, lat_b_degre:float,␣
 ↪lon_b_degre:float)-> tuple:

    R = 6378000 #Rayon de la terre en mètre

    lat_a = convertRad(lat_a_degre)
    lon_a = convertRad(lon_a_degre)
    lat_b = convertRad(lat_b_degre)
    lon_b = convertRad(lon_b_degre)

    dla = (lat_b - lat_a) / 2
    dlo = (lon_b - lon_a) / 2

    a = (sin(dla) * sin(dla)) + cos(lat_a) * cos(lat_b) * (sin(dlo) * sin(dlo))
    d = 2 * atan2(sqrt(a), sqrt(1 - a))

    return int(d * R)
```

1.3.2 Travail demandé

Proposez une fonction “estDansCercle” comprenant l’argument rayon en mètre et retourne un booléen True = **proche** ou False = **Eloigné**.

On donne deux trames :

trmGps1 = [“$GPGGA”, 134435.205, 4913.1500, “N”, 0357.5666,“E”, 1, “07”, 1.7, 711.3, “M”, 48.6, “M”, "“,”000*56"]

trmGps2 = [“$GPGGA”, 134434.159, 4914.8333, “N”, 0402.6666,“E”, 1, “07”, 1.9, 123.9, “M”, 53.2, “M”, "“,”000*12"]

Dans quelle ville se trouve le premier point ?

[26]:
```
#Ecrire votre fonction ici
# coding: utf-8

def estDansCercle(rayon = 0.0)->bool:

    # stockage de la trame 1
    trmGps1 = ["$GPGGA", 134435.205, 4913.1500, "N", 0357.5666,"E", 1, "07", 1.
↪7, 711.3, "M", 48.6, "M", "", "000*56"]

    # Stockage de la trame 2
    trmGps2 = ["$GPGGA", 134434.159, 4914.8333, "N", 0402.6666,"E", 1, "07", 1.
↪9, 123.9, "M", 53.2, "M", "", "000*12"]

    # Appel de la fonction decodeTrames pour stockage dans un tuple
    lat_a_degre, lon_a_degre, lat_b_degre, lon_b_degre = decodeTrames(trmGps1,␣
↪trmGps2)

    # Appel de la fonction distance et passage de la valeur dans distanceGps
    distanceGps = distance(lat_a_degre, lon_a_degre, lat_b_degre, lon_b_degre)

    # Affichage de la distance entre les deux points
    # ...

    # Returne True / Flase si la distance entre les deux coordonnées est dans␣
↪un rayon donné
    # ...
```

[27]:
```
# Testez votrefonction ici
# rayon en mètre
estDansCercle(6900)
```

```
La distance entre les deux points est de 6922 mètre(s).
```

[27]:
```
True
```

1.4 Les Réseaux sociaux

1.4.1 Autorisation

[22]:
```
# Executez cette fonction
# coding: utf-8

from datetime import date, datetime
#import datetime

def autorisation(dateNaissance:str)->int:
    # Calcul la date de ce jour
    today = date.today()
    # COnverti la date de naissance au format jj/mm/AAAA
    dateNaiss = datetime.strptime(dateNaissance, "%d/%m/%Y").date()

    # Calcul la différence entre ce jour et la date de naissance
    dif = (today - dateNaiss)

    # renvoi le nombre de seconde au total
    nbr = int(dif.total_seconds())

    return nbr
```

1.4.2 Travail demandé

Proposez une fonction “acces” demandant à l’utilisateur une chaine “date de naissance” et retourne un booléen True = accès autorisé ou False = Accès interdit.

Inspiré du manuel Delagrave Page 177

[31]:
```
#Ecrire votre fonction ici
# date au format jj/mm/aaaa
# coding: utf-8

def acces()->bool:

    # Demande la date de naissance de la personne
    votreAge = input("Donnez votre date de naissance au format jj/mm/aaaa : ")

    # Appel de la fonction autorisation
    age = autorisation(votreAge)

    # Converti l'age en année entière
    # ...

    # SI l'age est inférieur à 13 ans -> accès refusé
    # ...
```

[ ]:
```
# Testez votre fonction ici
acces()
```