Ikiam
Universidad Regional Amazónica

EXAMEN PARCIAL PYTHON

GBI6-2021II: BIOINFORMÁTICA

Apellidos, Nombres Flores Guerrero Belsabeth Juleth

03-08-2022

Sistema Operativo	Procesador	Ram
Windows 10	Intel(R) Core(TM) i7-8750H CPU @ 2.20GHz 2.21 GHz	8.00 GB

Color de texto

REQUERIMIENTOS PARA EL EXAMEN

Utilice de preferencia `Jupyter` de Anaconda, dado que tienen que hacer un control de cambios en cada pregunta.

Para este examen se requiere dos documentos:

1. Archivo `miningscience.py` donde tendrá dos funciones:
2. Archivo `2022I_GBI6_ExamenPython` donde se llamará las funciones y se obtendrá resultados.

Ejercicio 0 [0.5 puntos]

Realice cambios al cuaderno de jupyter:

- Agregue el logo de la Universidad
- Coloque sus datos personales
- Escriba una **tabla** con las características de su computador

Ejercicio 1 [2 puntos]

Cree el archivo `miningscience.py` con las siguientes dos funciones:

i. `download_pubmed` : para descargar la data de PubMed utilizando el **ENTREZ** de Biopython. El parámetro de entrada para la función es el `keyword` .

ii. `map_science` : para su data replique el ejemplo de MapOfScience (https://github.com/CSB-book/CSB/blob/master/regex/solutions/MapOfScience_solution.ipynb), donde los puntos resaltados son al menos 5 países

iii *Cree un docstring para cada función.*

Luego de crear las funciones, cargue el módulo `miningscience` como `msc` e **imprima docstring de cada función**.

In [10]:

```
# Escribadef download_pubmed (keyword):
    """
    Muestras de IDs de la busqueda en pubmed
    """
    from Bio import Entrez
    from Bio import SeqIO
    from Bio import GenBank
    Entrez.email = 'juleth.flores@est.ikiam.edu.ec'
    handle = Entrez.research(db='pubmed',
                         sort='relevance',
                         retmax='200',
                         retmode='xml',
                         term=keyword)
    results = Entrez.read(handle)
    id_list = results["IdList"]
    ids = ','.join(id_list)
    Entrez.email = 'juleth.flores@est.ikiam.edu.ec'
    handle = Entrez.efetch(db='pubmed',
                       retmode='xml',
                       id=ids)
    lista_id = ids.split(",")
    return (lista_id)


import csv
import re
import pandas as pd
from collections import Counter

def map_science(tipo):
    """ Docstring map_science """
    """ Esta funcion me permite crear un MapOfScience """
    #if tipo == "AD":
    with open() as f:
        my_text = f.read(tipo)
    my_text = re.sub(r'\n\s{6}', ' ', my_text)
    zipcodes = re.findall(r'[A-Z]{2}\s(\d{5}), USA', my_text)
    unique_zipcodes = list(set(zipcodes))
    unique_zipcodes.sort()
    unique_zipcodes[:10]
    zip_coordinates = {}
    with open('CSB-master/regex/data/MapOfScience/zipcodes_coordinates.txt') as f:
        csvr = csv.DictReader(f)
        for row in csvr:
         zip_coordinates[row['ZIP']] = [float(row['LAT']),
                                        float(row['LNG'])]
    zip_code = []
    zip_long = []
    zip_lat = []
    zip_count = []
    for z in unique_zipcodes:
    # if we can find the coordinates
        if z in zip_coordinates.keys():
            zip_code.append(z)
            zip_lat.append(zip_coordinates[z][0])
            zip_long.append(zip_coordinates[z][1])
            zip_count.append(zipcodes.count(z))
    import matplotlib.pyplot as plt
    #%matplotlib inline
```

```
    plt.scatter(zip_long, zip_lat, s = zip_count, c= zip_count)
    plt.colorbar()
# only continental us without Alaska
    plt.xlim(-125,-65)
    plt.ylim(23, 50)
# add a few cities for reference (optional)
    ard = dict(arrowstyle="->")
    plt.annotate('Tokio', xy = (-122.1381, 37.4292),
                 xytext = (-112.1381, 37.4292), arrowprops= ard)
    plt.annotate('Seúl', xy = (-71.1106, 42.3736),
                 xytext = (-73.1106, 48.3736), arrowprops= ard)
    plt.annotate('Paris', xy = (-87.6847, 41.8369),
                 xytext = (-87.6847, 46.8369), arrowprops= ard)
    plt.annotate('Osaka', xy = (-122.33, 47.61),
                 xytext = (-116.33, 47.61), arrowprops= ard)
    plt.annotate('Londres', xy = (-80.21, 25.7753),
                 xytext = (-80.21, 30.7753), arrowprops= ard)
    params = plt.gcf()
    plSize = params.get_size_inches()
    params.set_size_inches( (plSize[0] * 3, plSize[1] * 3) )
```

```
  File "C:\Users\aula\AppData\Local\Temp/ipykernel_13572/1403090939.py", lin
e 2
    """
    ^
IndentationError: unexpected indent
```

Ejercicio 2 [2 puntos]

Utilice dos veces la función `download_pubmed` para:

- Descargar la data, utilizando los keyword de su preferencia.
- Guardar el archivo descargado en la carpeta `data` .

Para cada corrida, imprima lo siguiente:

```
'El número artículos para KEYWORD es: XX' # Que se cargue con inserción de texto o
valor que correspondea KEYWORD y XX
```

In [2]:

```
# Escriba aquí su código para el ejercicio 2
```

Ejercicio 3 [1.5 puntos]

Utilice dos veces la función `map_science` para:

- Visualizar un mapa para cada data descargada en el ejercicio 2.
- Guardar los mapas en la carpeta `img`

In [4]:

```
# Escriba aquí su código para el ejercicio 3
```