EXAMEN PARCIAL PYTHON

GBI6-2021II: BIOINFORMÁTICA

Apellidos, Nombres <--- CAMBIE POR LOS QUE CORRESPONDA A SUS DATOS

03-08-2022

Color de texto

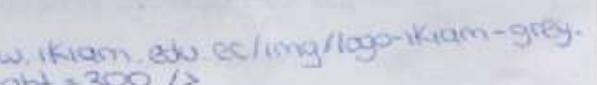
REQUERIMIENTOS PARA EL EXAMEN

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:

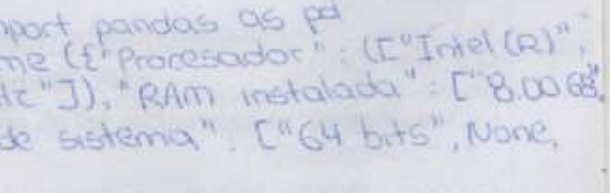
1. Archivo `miningscience.py` donde tendrá dos funciones:
2. Archivo `2022I_GBI6_ExamenPython` donde se llamará las funciones y se obtendrá resultados.

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. `science_plots` : la función debe

- utilizar como argumento de entrada la data descargada por `download_pubmed`
- ordenar los conteos de autores por país en orden ascedente y
- seleccionar los cinco más abundantes. Con esta selección debe graficar un `pie_plot` . Como guía para el conteo por países puede usar el ejemplo de MapOfScience (https://github.com/CSB-book/CSB/blob/master/regex/solutions/MapOfScience_solution.ipynb).

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 [1]:

```
# Escriba aquí su código para el ejercicio 1
Import miningscience_g01 as msc
help (msc.download_pubmed)
help (msc.science_plots)
```

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
Import os
Import re
x = msc.download_pubmed ("Dogs")
y = len (x)
print ("El número articulos para KEYWORD es: ", y)
with open ("../Data/Dogs.txt", "w") as txt:
  txt.write (x)
j = msc.download_pubmed ("virus")
k = len (j)
print ("El número articulos para KEYWORD es: ", k)
with open ("../Data/virus.txt", "w") as txt:
  txt.write (j)
```

Ejercicio 3 [1.5 puntos]

Utilice dos veces la función `science_plots` para:

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

In [4]:

```
# Escriba aquí su código para el ejercicio 3
Dogs = msc.science_plots(x)
with open("../img/Dogs.JPG","w") as JPG:
    Dogs.

Virus = msc.science_plots(j)
with open("../img/Virus.JPG","w") as JPG:
    Virus.
```

Ejercicio 4 [1 punto]

Interprete los resultados de las figuras del **ejercicio 3**

Escriba la respuesta del ejercicio 5 .

- Los cinco países en los que se cuenta con más autores que han realizado investigaciones con la keyword "Dogs" son Japón, Italia, China, Brazil y Alemania.
- Los cinco países en los que se cuenta con más autores que han realizado investigaciones con la keyword "Virus" son China, Francia, Alemania, Australia y Brazil.

Ejercicio 5 [2 puntos]

Para algún **gen de las enzimas que intervienen en la ruta metabolica de la gluconeogenesis** (Lista de genes por tipología (https://www.genome.jp/pathway/map00010+C00068)), realice lo siguiente:

1. Una búsqueda en la página del NCBI nucleotide (https://www.ncbi.nlm.nih.gov/nucleotide/).
2. Descargue el `Accession List` de su búsqueda y guarde en la carpeta `data` .
3. Cargue el `Accession List` en este notebook y haga una descarga de las secuencias de los **quince primeros** IDs de la accesión.
4. Arme un árbol filogenético para los resultados del paso 3.
5. Guarde su arbol filogénetico en la carpeta `img`
6. Interprete el árbol del paso 4.

In [3]:

```
# Escriba aquí su código para el ejercicio 6
From Bio import Entrez
From Bio import Phylo
import matplotlib.pyplot as plt
From Bio.Phylo.TreeConstruction import DistanceTreeConstructor
from Bio import SeqIO
From Bio.phylo.TreeConstruction import DistanceCalculator
From Bio.Align.Applications import ClustalwCommandline
From Bio import AlignIO
import os
with open ("../data/sequence.seq") as file:
    text = File.read()
text = text.split('\n')
text = '\n'.join(text[:15])
handle = Entrez.efetch(db="nucleotide", rettype="gb", retmode="txt",
id = txt)
print(hande.url)
records = SeqIO.parse("../data/sequence.gb",
"fasta") "genbank")
Count = SeqIO.write(records, "../data/sequence.
Fasta", "fasta")
Clustalw_exe = r"C:/program
Files (x86)/Clustal w2/clustal
w2.exe"
Clustalw_cline = ClustalwCommand
line(clustalw_exe, infile = "../
data/sequence.fasta")
assert os.path.isfile
(clustalw_exe), infile
= "../data/sequence.
Fasta")
stdout, stderr = clustalw_cline()
print(clustalw_cline)
ClustalAlign = AlignIO.read(aln,
"clustal")
Calculator = DistanceCalculator("iden-
tity")
distance_matrix = calculator.get_dista
ce(alignment)
Constructor = DistanceTreeConstructor(Calculator)
tree = constructor.build_tree(alignment)
tree = rooted = True
Fig = plt.figure(figsize=(10,12), dpi=200)
plt.rc("font", size=12)
plt.rc("xtick", labelsize=20)
plt.rc("ytick", labelsize=20)
axes = Fig.add_subplot(1,1,1)
phylo.draw(tree, axes=axes)
Fig.savefig("../img/arbolfilogenético.jpg")
```

Escriba aquí la interpretación del árbol

Descripción:
Inner 1 es antecesor de XM_029014683.1 y XM-039427779.1, Inner 2 es antecesor de Inner 1 y XM-024718619.1, Inner 3 es antecesor de Inner 2 y XM-040253125.1

Ejercicio 6 [1 punto]

1. Cree en GitHub un repositorio de nombre `GBI6_ExamenPython`.
2. Cree un archivo `Readme.md` que debe tener lo siguiente:

- Datos personales
- Características del computador
- Versión de Python/Anaconda y de cada uno de los módulos/paquetes y utilizados
- Explicación de la data utilizada
- Un diagrama de procesos del módulo `miningscience`

3. Asegurarse que su repositorio tiene las carpetas `data` e `img` con los archivos que ha ido guardando en las preguntas anteriores.
4. Realice al menos 1 control de la versión (commits) por cada ejercicio (del 1 al 5), con un mensaje que inicie como:

```
Carlitos Alimaña ha realizado el ejercicio 1

Carlitos Alimaña ha realizado el ejercicio 2

...
```

```python
# NOMBRE (Elizalde, Jennyffer):

from Bio import Entrez
from Bio import SeqIO
from Bio import GenBank
import re
import pandas as pd
import matplotlib.pyplot as plt
import csv as csv

def download_pubmed (keyword):
    """
    Esta función permite la búsqueda científica en la base de datos Pubmed
    """

    Entrez.email = "A.N.Other@example.com"
    busq = Entrez.read(Entrez.esearch(db="pubmed",
                            term=keyword,
                            usehistory="y"))
    webenv = busq["WebEnv"]
    query_key = busq["QueryKey"]
    handle = Entrez.efetch(db="pubmed",
                           rettype="medline",
                           retmode="text",
                           retstart=0,
                           retmax=543, webenv=webenv, query_key=query_key)
    data = handle.read()
    dataexp = re.sub(r'\n\s{6}','', data)
    return dataexp


def science_plots(archivo):
    """
    Esta función permite ordenar ascendentemente el número de autores que realizaron un estudio científico de acuerdo a la búsqueda en la
función 1 y seleccionar los 5 más abundantes
    """
    a = re.sub(r'\s[\w._%+-]+@[\w.-]+\.[a-zA-Z]{2,4}','',archivo)
    b = re.sub(r'\..\d.\.','.',a)
    c = re.sub(r'\..\d.','',b)
    x=c[1:].split('PMID-')

    Paises1=[]
    for PMID in x:
        q=PMID.split('\n')
        for fila in q:
            w=fila.split(' ')
            if w[0] == 'AD':
                e=fila.split(',')
                Paises1.append(e[-1])

    p=0
    Paises2 =[0]*len(Paises1)
    for lls in Paises1:
        bytes(lls,encoding="utf8")
        if lls != '':
            w=lls
            if w[0] == ' ':
                w = re.sub (r'^\s','',w)
            if w[-1] == '.':
                w = re.sub (r'\.$','',w)
            w = re.sub (r'\.$','',w)
            w = re.sub (r'\s$','',w)
        Paises2[p]=w
        p=p+1
```

```
    Pulses_completos=['Andorra','United Arab Emirates ','Afghanistan','Antigua and Barbuda','Anguilla','Albania','Armenia','Netherlands
Antilles','Angola','Antarctica','Argentina','American Samoa','Austria','Australia','Aruba','Azerbaijan','Bosnia and
Herzegovina','Barbados','Bangladesh','Belgium','Burkina Faso','Bulgaria','Bahrain', 'Burundi','Benin','Bermuda','Brunei','Bolivia',
'Brazil', 'Bahamas','Bhutan','Bouvet Island','Botswana','Belarus','Belize','Canada','Cocos [Keeling] Islands','Congo [DRC]','Central
African Republic','Congo Republic', 'Switzerland','Côte d'Ivoire','Cook Islands','Chile','Cameroon','China','Colombia','Costa
Rica','Cuba', 'Cape Verde','Christmas Island','Cyprus','Czech Republic','Germany','Djibouti','Denmark','Dominica','Dominican
Republic','Algeria','Ecuador' ,'Estonia','Egypt','Western Sahara','Eritrea','Spain','Ethiopia','Finland','Fiji','Falkland Islands [Islas
Malvinas]','Micronesia','Faroe Islands','France','Gabon', 'United Kingdom','Grenada','Georgia','French
Guiana','Guernsey','Ghana','Gibraltar','Greenland','Gambia', 'Guinea','Guadeloupe','Equatorial Guinea','Greece','South Georgia and the
South Sandwich Islands','Guatemala','Guam','Guinea-Bissau','Guyana','Gaza Strip','Hong Kong','Heard Island and McDonald
Islands','Honduras','Croatia', 'Haiti','Hungary','Indonesia','Ireland' ,'Israel','Isle of Man','India','British Indian Ocean
Territory','Iraq', 'Iran','Iceland','Italy','Jersey','Jamaica','Jordan',
'Japan','Kenya','Kyrgyzstan','Cambodia','Kiribati','Comoros','Saint Kitts and Nevis','North Korea','South Korea','Kuwait','Cayman
Islands','Kazakhstan','Laos','Lebanon','Saint Lucia','Liechtenstein','Sri Lanka','Liberia','Lesotho','Lithuania','Luxembourg','Latvia'
,'Libya','Morocco','Monaco','Moldova','Montenegro','Madagascar','Marshall Islands','Macedonia [FYROM]','Mali','Myanmar
[Burma]','Mongolia' ,'Macau','Northern Mariana
Islands','Martinique','Mauritania','Montserrat','Malta','Mauritius','Maldives','Malawi','Mexico','Malaysia' ,'Mozambique','Namibia','New
Caledonia','Niger','Norfolk Island','Nigeria','Nicaragua','The Netherlands','Norway','Nepal','Nauru', 'Niue','New
Zealand','Oman','Panama','Peru','French Polynesia', 'Papua New Guinea','Philippines','Pakistan','Poland','Saint Pierre and Miquelon'
,'Pitcairn Islands','Puerto Rico','Palestinian Territories','Portugal','Palau','Paraguay','Qatar','Reunion','Romania', 'Serbia','Russia'
,'Rwanda','Saudi Arabia', Solomon Islands','Seychelles','Sudan','Sweden','Singapore','Saint Helena','Slovenia', 'Svalbard and Jan
Mayen','Slovakia','Sierra Leone','San Marino','Senegal','Somalia','Suriname','Sao Tome and Principe','El Salvador','Syria', 'Swaziland'
,'Turks and Caicos Islands','Chad','French Southern Territories','Togo','Thailand','Tajikistan','Tokelau','Timor-Leste','Turkmenistan'
,'Tunisia','Tonga','Turkey','Trinidad and Tobago','Tuvalu','Taiwan','Tanzania','Ukraine','Uganda','U.S. Minor Outlying Islands','United
States of America','Uruguay','Uzbekistan','Vatican City','Saint Vincent and the Grenadines','Venezuela', 'British Virgin Islands','U.S.
Virgin Islands','Vietnam','Vanuatu','Wallis and Futuna','Samoa','Kosovo','Yemen','Mayotte','South Africa','Zambia','Zimbabwe']
    Pulses3=Pulses2
    h=Pulses_completos
    #elen(h)
    CountriesCount=[0]*9
    k=0
    for elem in h:
        d=0
        for comp in Pulses3:
            if elem == str(comp):
                d=d+1
        CountriesCount[k]=d
        k=k+1

    Pulses4=[]
    Counter=[]
    c=0
    for line in CountriesCount:
        if str(line) != '0':
            Counter.append(line)
            su=Pulses_completos[c]
            Pulses4.append(n)
        c=c+1

    Table1 = pd.DataFrame({'Country' : Pulses4,
                           'num_auth' : Counter})
    Order=Table1.sort_values(by['num_auth'], ascending=[False])
    Taken = Order.iloc[0:5]
    sum=Taken['num_auth'].sum()
    lo= Taken.iloc[:,0]
    sum=pd.Series(lo)
    l = Taken.iloc[:,1]
    sa = pd.Series(l)
    s = sa.tolist()

    prun = []
    for number in s:
        xa=(number/sum)*100
        prun.append(xa)
    Table2 = pd.DataFrame({'Country' : su,
                           'Percent' : prun})

    fig1, ax1 = plt.subplots()
    ax1.pie(prun, labels=su, autopct='%1.1f%%',
            shadow=True, startangle=90)
    ax1.axis('equal')
    plt.show()


    return (Table2)
```