Examen Parcial 2 Python.

Escrito del Examen.

Modulo para los ejercicios 1, 2, 3, 4.

Carga de librerías.

```
From Bio import Entrez
import re
from IPython.core.display import Image
from Bio.Seq import Seq
from Bio.SeqUtils import GC
from reportlab.lib import colors
from reportlab.lib.units import cm
from Bio.Graphics import GenomeDiagram
from Bio import SeqIO
from Bio.SeqFeature import SeqFeature, FeatureLocation
import csv
import pandas as pd
from collections import Counter
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.mlab as mlab
import math
import seaborn as sns.
```

Scanned by TapScanner

Funciones.

```
def download_pubmed(keyword):
    """ Función que extrae listado de artículos desde pubmed a través de un keyword
    entre comillos """
    Entrez.email = "ronald.bazantes@est.iklam.edu.ec"
    Entr=Entrez.read(Entrez.esearch(db="pubmed",
                                    term="Cancer"          ---> Cambia el keyword
                                    usehistory="y")))      para el ejercicio 2

    webenv=Entr["WebEnv"]
    query_key=Entr["QueryKey"]
    handl=Entrez.efetch(db="pubmed",
                        rettype='medline',
                        retmode="text",
                        retstart=0,
                        retmax=543, webenv=webenv, query_key=query_key)
    out_handl = open("data/Cancer_pubs.txt", "w")
    m=handl.read()
    out_handl.write(m)
    out_handl.close()
    handl.close()
    retur m.
```

Scanned by TapScanner

```
def seaborn_plot(keyword):
    """Esta función nos ayuda a extraer datos según se indique el keyword
    en relación a las mismas palabras usadas en el ejercicio 2, pues de esta manera
    se realizó una función en general."""
    with open("datos/Cancer_pubs.txt", errors="ignore") as l:
        texto = l.read()  <- cambiar el nombre para el ejercicio 2.
    texto = re.sub(r"\n\s{6}", " ", texto)
    countries_1 = re.findall(r"AD\s{2}-\s[A-Za-z].*", \s([A-Za-z]
                             *)\.\s", texto)

    unique_countries = list(set(countries_1)).
    conteo = Counter(countries_1)
    resultado = {}
    for clave in conteo:
        valor = conteo[clave]
        if valor > 1:
            resultado[clave] = valor
    ordenar = (sorted(resultado.values()))
    ordenar.sort(reverse = True)
    import operator
    pais = []
    contador = [].

    reverse = sorted(resultado.items(), key = operator.itemgetter(1), reverse=True)
    for name in enumerate(reverse):
        pais.append(name[1][0])
        contador.append(resultado[name[1][0]])
    paises_top = pais[0:5]
```

Scanned by TapScanner

```
frecuencia_cinco = contador[0:5]
fig = plt.figure(figsize=(10, 7))
plt.pie(frecuencia_cinco, labels = paises_top)
(plt.savefig("img/Cancer_pubs.png", dpi=200, bbox_inches='tight'))
plt.show()
```

Si cambiar el nombre para el ejercicio 2.

```
return plt.show
```

Ejercicio 0.

```
import pandas
```

```
<tables
  <tr>
    <th> Procesador </th>
    <th> RAM </th>
    <th> Tipo de Sistema </th>
    <th> Edición </th>
    <th> Versión </th>
  </tr>
  <tr>
    <td> Intel (R) Core (TM) i7-10750H CPU @2.60GHz 2.59GHz </td>
    <td> 8GB </td>
    <td> Sistema operativo de 64 bits </td>
    <td> Windows 11 Pro </td>
    <td> 21H2 </td>
  </tr>
</table>
```

Scanned by TapScanner

```
! pip install bio python
! pip install reportlab
! pip install nglview

Ejercicio 1.
1.1.

from Bio import Entrez.
import re
from IPython.core.display import Image.
from Bio.Seq import Seq
from Bio.Seq Utils import GC
from reportlab.lib import colors.
from reportlab.lib.units import cm.
from Bio.Graphics import GenomeDiagram.
from Bio import SeqIO.
From Bio.SeqFeature import SeqFeature, FeatureLocation.
import miningscience as msc.
from msc import download_pubmed (keyword).
help (download_pubmed)
From msc import science_plot (keyword)
help (science_plot.

Ejercicio 2.
from msc import download_pubmed21.
from msc import download_pubmed22.
Ecoli = download_pubmed21("Escherichia coli")
Sp_ven = (download_pubmed22 ("Spider venom"))
```

Scanned by TapScanner

print ("El número artículos para Escherichia coli es", len(E_coli))
print ("El número artículos para Spider venom es", len(Sp_ven)).

Ejercicio 3.

```
from nsc import science_plot21
from nsc import science_plot22.
science_plot21 ("Escherichia coli")
science_plot22 ("Spider venom")
```

Ejercicio 5.

```
from Bio import Phylo
from Bio.Phylo.TreeConstruction import DistanceCalculator.
from Bio.Phylo.TreeConstruction. import DistanceTreeConstructor.
from Bio import AlignIO.
from Bio import SeqIO
from Bio import Entrez.
import re
import os.
from Bio.Align.Applications import ClustalwCommandline
with open("data/sequence.seq") as f:
    data = f.readlines()[0:15].
out_sequence = open("data/sequence.fasta", "w")
for lines in data:
    Entrez.email = "ronald.brenes@est.itcr.ac.cr"
    handle = Entrez.efetch(db = "nucleotide", id=lines, rettype="fasta", retmode=
                                                                    "text")
    data = (handle.read())
    out_sequence.write(data)
```

Scanned by TapScanner

```
out_sequence.close()
From Bio.Align.Applications import ClustalwCommandline.
clustalw_exe = r"E:\Cosas de Ronald\Trabajos y Deberes\BioInformática\Python\
                ClustalW2\ClustalW2\clustalw2.exe"
clustalw_cline = ClustalwCommandline(clustalw_exe, infile = "data/sequence.fasta")
assert os.path.isfile(clustalw_exe), "Clustal_W executable is missing or not found?"
stdout, stderr = clustalw_cline()
print(clustalw_cline)
ClustalAlign = AlignIO.read("sequence.aln", "clustal")
print(ClustalAlign)
from Bio import Phylo.
tree = Phylo.read("sequence.dnd", "newick").
with open("data/sequence.aln", "r") as aln:
    alignment = AlignIO.read(aln, "clustal")
from Bio.Phylo.TreeConstruction import DistanceCalculator
calculator = DistanceCalculator('identity')

distance_matrix = calculator.get_distance(alignment).

from Bio.Phylo.TreeConstruction import DistanceTreeConstructor

constructor = DistanceTreeConstructor(calculator).

mic_tree = constructor.build_tree(alignment).

mic_tree.rooted = True

Phylo.write(mic_tree, "tree.xml", "phyloxml").

mic_tree = Phylo.read(files="tree.xml", format="phyloxml")

import matplotlib.

import matplotlib.pyplot as plt.
```

Scanned by TapScanner

```
fig = plt.figure(figsize=(30,40), dpi=100)
mat plotlib.rc('font', size=30)
matplotlib.rc('xtick', labelsize=20)
matplotlib.rc('ytick', labelsize=20)
axes = fig.add_subplot(1,1,1)
Phylo.draw(mic_tree, axes=axes)
fig.savefig("img/Phylogen_tree.jpg")
```

Scanned by TapScanner