Guía 3

May 10, 2024

1 Guía de Trabajos Prácticos

Los contenidos de esta unidad son: - Librerias. Uso de librerias. - Funciones. Argumentos. Variables locales y globales,args and kwargs - Desarrollo de librerias propias. - Clases y objetos.

En algunos de los problemas a continuación se les pedirá que completen código en los lugares marcados con la palabra **TODO**.

1.1 Problema 1: Producto de Matrices

Implemente una función llamada *mat_mul* que calcule el resultado del producto de dos matrices A y B. La función debe contar con una mínima documentación y debe testear que A y B tengan dimensiones compatibles. El script tiene que llamarse *mat_mul.py*

1.2 Problema 2: Factorial

El factorial de un número entero no negativo n, denotado como $n!$, se define como el producto de todos los enteros positivos menores o iguales a n,

$$n! = n \times (n-1) \times (n-2) \times \ldots \times 1, \qquad \text{con la convención de que } 0! = 1.$$

Implemente dos versiones del cálculo del factorial: (i) la versión recursiva (que usa invocaciones a la misma función) y (ii) la iterativa (que usa un *for* para el cálculo).

1.2.1 Codigo

[1]:
```
%%writefile factorial.py

def factorial(lista,metodo='iterativo'):
    """  TODO   """
    if   metodo=='iterativo':
        #TODO
        pass
    elif tipo=='recursivo':
        #TODO
        pass
    else:
        #TODO
        pass
```

```
Overwriting factorial.py
```

[2]:
```
%run factorial
```

1.3 Problema 3: Sequencia de Fobonaci

1. Escriba una función que permita calcular el n-ésimo término en la secuencia de Fibonacci usando recursión.
2. Introduzca una variable global para llevar cuenta del número de llamadas se hicieron en el cálculo recursivo.

1.3.1 Codigo

[3]:
```
%%writefile fibonacci.py

def fibonacci(n):
    """  TODO   """
    pass
```

```
Overwriting fibonacci.py
```

[4]:
```
%run fibonacci
```

1.3.2 Test

[5]:
```
# 1
print('El n=15 elemento de la secuencia de Fibonacci es: ',fibonacci(15))
```

```
El n=15 elemento de la secuencia de Fibonacci es:  None
```

[6]:
```
# 2
global ncalls
ncalls = 0
print('El n=15 elemento de la secuencia de Fibonacci es: ',fibonacci(15))
print('Se hicieron ',ncalls,' llamadas recursivas.')
```

```
El n=15 elemento de la secuencia de Fibonacci es:  None
Se hicieron  0  llamadas recursivas.
```

1.4 Problema 4: Medidas de tendencia central

En el primer *TODO* tienen que completar la documentación de la función y, en los tres siguientes, tienen que implementar la media, mediana y moda de una lista de datos llamada *lista*.

1.4.1 Codigo

[7]:
```
%%writefile tendencia_central.py

def tendencia_central(lista,tipo='media'):
    """  TODO   """
    if   tipo=='media':
        #TODO
        pass
    elif tipo=='mediana':
        #TODO
        pass
    elif tipo=='moda':
        #TODO
        pass
    else:
        #TODO
        pass
```

```
Overwriting tendencia_central.py
```

[8]:
```
%run tendencia_central
```

1.4.2 Test

Testee el funcionamiento de la función con las notas de una prueba de matemática de niños de 6 grado

[9]:
```
notas=[7,7,6,8,3,4,9,3,7,3,8,5,9,7,4,8,8,7,4,7,6,7,8,4,4,7,3,5,7,8,5]
```

[10]:
```
import statistics as st
media, mediana, moda = tendencia_central(notas,tipo='media'),␣
 ↪tendencia_central(notas,tipo='mediana'), tendencia_central(notas,tipo='moda')

if media == st.mean(notas) and mediana == st.median(notas) and moda == st.
 ↪mode(notas):
    print('La media, la mediana y la moda de las notas son: ',media,mediana,moda)

else:
    print('Funciona la media mediana moda?:',media == st.mean(notas),mediana ==␣
 ↪st.median(notas),moda == st.mode(notas))
```

```
Funciona la media mediana moda?: False False False
```

1.5 Problema 5: Ordenando enteros... 8238075123

Vamos a ordenar una lista de enteros, por ejemplo podríamos ordenar los dígitos del dni para que aparezcan de menor a mayor usando el algoritmo de ordenamiento de burbuja. Nuestro código del

algoritmos de burbuja tiene que comparar dígitos contiguos y reordenarlos. Hay que hacer tantas pasadas a travez de la lista como sea necesario.

1.5.1 Codigo

[11]:
```
%%writefile burbuja.py

def burbuja(seq):
    """ TODO """

    # TODO
```

```
Overwriting burbuja.py
```

[12]:
```
%run burbuja
```

1.5.2 Test

Testee la rutina con su número de dni

[13]:
```
dni = '39641192'
dni = list( map(int,dni) )
print(dni)
```

```
[3, 9, 6, 4, 1, 1, 9, 2]
```

[14]:
```
burbuja(dni)
```

[15]:
```
dni = ''.join(map(str,dni) )
dni
```

[15]:
```
'39641192'
```

1.6 Problema 6: Ordenando palabras... *gato, perro, pájaro, pez*

Debe implementar una función que tenga como *input* una lista de palabras y cómo *output* las mismas palabras pero ordenadas alfabéticamente. Esta función debe incluirse como parte de un programa que le solicite las palabras a ordenar al usuario, elimine de la lista de palabras las duplicadas, y le imprima en pantalla la lista de palabras ordenadas.

1.7 Problema 7: ¿Es palíndromo?

Según la RAE:

Palíndromo: Palabra o frase cuyas letras están dispuestas de tal manera que resulta la misma leída de izquierda a derecha que de derecha a izquierda; p. ej., *anilina*; *dábale arroz a la zorra el abad.*

Escriba un script que implemente una función llamada *espalindromo* que tome una palabra como entrada y determine si es un palíndromo. La función debe devolver *True* si la palabra es un palíndromo y *False* en caso contrario.

Descripción: la función debe tener un primer paso en el cual se transforme la palabra/frase para que se ignore mayúsculas/minúsculas, espacios en blanco, signos de puntuación números y otros caracteres, los caracteres permitidos deben ser *'abcdefghijklmnñopqrstuvwxyz'*, es decir, por ejemplo, si se ingresa _string='#Ho0la.*'_ luego del primer paso debemos tener *string='hola'*; y un segundo paso donde se haga la comparación de los caracteres correspondientes. El segundo paso puede hacerse de manera recursiva o iterativa.

1.7.1 Codigo

[16]:
```
%%writefile espalindromo.py

def espalindromo(string):
    """ TODO """
    #Paso 1: uniformación
    # TODO

    #Paso 2: comparación
    # TODO

    pass
```

```
Overwriting espalindromo.py
```

[17]:
```
%run espalindromo
```

1.7.2 Test

[18]:
```
print(espalindromo("anilina"))              # Debe imprimir: True
print(espalindromo("RomA"))                 # Debe imprimir: False
print(espalindromo("Anita lava la tina"))   # Debe imprimir: True
```

```
None
None
None
```

1.8 Problema 8: MCD

El algoritmo de Euclides permite calcular el máximo común divisor (MCD) de dos números enteros m y n. El algoritmo se detalla a continuación. Sean m, n dos números naturales tal que $m > n$, para encontrar su máximo común divisor: - Dividir m por n para obtener el resto r donde $0 \leq r < n$. - Si $r = 0$, el MCD es n. - Si no, el máximo común divisor es $MCD(n, r)$.

Implementar una función que calcule el MCD.

1.8.1 Codigo

[19]:
```
%%writefile mcd.py

def mcd(m,n,metodo='iterativo'):
    """  TODO   """
    if   metodo=='iterativo':
        #TODO
        pass
    elif tipo=='recursivo':
        #TODO
        pass
    else:
        #TODO
        pass
```

```
Overwriting mcd.py
```

[20]:
```
%run mcd
```

1.9 Problema 9: Personas

Vamos a crear una clase llamada *Persona* que nos va a servir para almacenar algunos atributos y algunas opearaciones. Esta clase debe contar con los siguientes métodos: - *setBirthday* define la fecha de cumpleaños en el formato *datetime*.

- *getLastName* devuelve el apellido de la persona.
- *getAge* devuelve la edad en días, si está definida.
- _ _ gt _ _(self,B) que define si la persona self es mayor que la persona B.
- _ _ str _ _ devuelve el nombre

```
Overwriting persona.py
```

[22]:
```
%run persona.py
```

[ ]:
```

```