relatoría 2

October 23, 2022

1 RELATORIA 2 INFORMATICA 2

1.1 OBJETIVOS

1. aprender sobre el funcionamiento de los ciclos while y for
2. aprender a usar de manera apropiada y efectiva los métodos de los strings, las listas, los diccionarios y las tuplas
3. identificar y crear funciones, módulos y clases, esto usándolos para hacer un medio de trabajo más ordenado

1.2 06-09-2022

1.2.1 CICLO WHILE

Tiene similitud con la sentencia if. Sin embargo, la condición puede cambiar.

Si la condición es verdadera, se ejecutan todas las sentencias que contenga el ciclo while. Si la condición es falsa, no se ejecutan las sentencias contenidas

if condición:

while :

. . . ..

while :

else:

Debemos tener en cuenta que un ciclo while puede volverse infinito, lo cual es un problema para la programación por tanto es mejor intentar no usarlo mucho, aunque sigue siendo muy util cuando se esta empezando a programar, o cuando es un programa sencillo

podemos usar el ciclo while como contador o como iterable

algunos ejemplos son:

[ ]:
```
# 06/09/2022


#Desarrolle un ciclo while infinito

# condicion = True
#
# while condicion:
#     print("ciclo ejecutado")

#Cómo protegernos de un ciclo infinito

condicion = True
contador = 0
while condicion:
     print("ciclo ejecutado {}".format(contador))
     contador = contador + 1
     if contador == 100:
        break


#Realice un ciclo while con un numero secreto. Cada vez que se ejecuta
#un ciclo, el programa pide adivinar el numero, en caso de no ser acertado
#se debe mostrar un mensaje diciendo: "Estás atrapado". Y en caso contrario
#terminar el ciclo y avisar que el numero es correcto.

#numero_secreto = 999
#numero_usuario = int(input("Ingrese el número secreto: "))
#condicion = (numero_secreto != numero_usuario)
#
#while condicion:
#    print("No es el número correcto")


#Realice un ciclo while que imprima los números del 10 al 100, separados por␣
 ↪guion(-)
#sin salto de linea
#10 - 11 - 12 - 13 - ... 100

contador=10
lista= []
while contador<=100 :
        lista.append(contador)     #la funcion append(), que toma la lista y le␣
 ↪agrega valores
```

```
        contador=contador+1
print(str(lista))

contador= 11
StrNumeros= "10"

while contador<=100 :

        StrNumeros= StrNumeros + "-"+ str(contador)
        contador= contador+1
print(StrNumeros)

# mostrar  en pantalla los numeros del 200 al 100 utilizando ciclo while sin␣
 ↪salto de linea

contador =200
while contador>=100:
        print(contador, end=" ")
        contador= contador-1

# mostrar  en pantalla los numeros del 200 al 100 utilizando ciclo while sin␣
 ↪salto de linea

contador = 200

while contador >=100 :
        if contador%10==0:
                print(contador)
                contador= contador-1
        else:
                print(contador,end=" ")
                contador= contador-1


# pida a un usuario que ingrese numeros, en caso que asi lo desee. de los␣
 ↪numeros calcule
# cual es el mayor de los ingresados. y si el usuario no quiere ingresar mas␣
 ↪numeros
# el ciclo debe terminar
```

1.3 08-09-2022

1.3.1 CICLO FOR

En este caso no es necesario evaluar una condición en un ciclo (tal como sucede con el while). Sabemos lo que vamos a recorrer (un iterable), por lo tanto utilizamos la sentencia for

En python, el ciclo for se utiliza de una manera muy diferente a otros lenguajes, lo que se hace es

recorrer un ITERABLE.

———notación ————————-

for in : . . . . . .

=> puede ser un nombre cualquiera elegido por el programador => Cualquier cosa que se pueda recorrer. Cadenas, listas, tuplas, arreglos

ejemplo:

cadena = “HolaMundoCruel” lista = [1,2, 30, 100, 50, -20] tupla = (1,2,3,1,2,3,1,2,3) resultados = [1,-1,1,-1,1,-1] rango = range(1,100)

for carácter in cadena: print(carácter, end=“–”)

for i in lista: print(i, end= “–”)

for numero in tupla: print(numero, end= “–”)

for resultado in resultados: print(resultado, end= “–”)

for i in rango: print(i, end=“–”)

El ciclo for es mas utilizado por los programadores ya que es el iterable por excelencia, en un ciclo while es muy fácil crear un ciclo infinito que nos dañe el programa, con el ciclo for no pasa eso. simplemente se corren cosas finitas lo que nos salva de muchos desastres computacionales

los ciclos for y while ameritan mucha memoria RAM, por eso en un sistema de analitica de datos o de simulación de muchas particulas son un problema. debemos buscar otras alternativas

algunos ejemplos son:

[ ]:
```
#08-09-2022

"""Recorrer los siguientes iterables"""
cadena = "HolaMundoCruel"
lista = [1,2, 30, 100, 50, -20]
tupla = (1,2,3,1,2,3,1,2,3)
resultados = [1,-1,1,-1,1,-1]
rango = range(1,100)

print("Recorrido de cadena  ==> ")
for caracter in cadena:
    print(caracter, end="--")

print("\n\nRecorrido de lista ==> ")
for i in lista:
    print(i, end= "--")

print("\n\nRecorrido de tupla ==> ")
for numero in tupla:
    print(numero, end= "--")
```

```
print("\n\nRecorrido de resultados  ==> ")
for resultado in resultados:
    print(resultado, end= "  ")

print("\n\nRecorrido de rango  ==>")
for i in rango:
    print(i, end="--")

"""
Recorrer los números  del 1 al 10 utilizando diferentes iterables (por lo menos␣
 ↪4)
sin necesidad de definirlos en una variable
"""

#forma 1
print("\n\nForma 1 =>")
for i in [1,2,3,4,5,6,7,8,9,10]:
    print(i, end= "  ")

#forma 2
print("\nForma 2 =>")
for i in range(1,11):
    print(i, end= "  ")

#forma 3
print("\nForma 3 =>")
for i in (1,2,3,4,5,6,7,8,9,10):
    print(i, end= "  ")

#forma 4
print("\nForma 4 =>")
for i in "12345678910":
    print(i, end = "  ")

"""
 Imprimir los numeros pares del 0 al 20
"""
print("\n\n Numeros pares")
for i in range(0, 21, 2):
    print(i, end=" ")


"""
Imprimir los números multiplos de 4 desde el 5 hasta 30
"""

print("\n\n Opción 1 numeros multiplos de 4")
```

```
for i in range (8,29,4):
    print (i, end="   ")

print("\n\n Opción 2 numeros multiplos de 4")
for m in range(5,31):
    if m%4 == 0:
        print(m)
```

1.4 13-09-2022

1.4.1 ITERABLES Y MÉTODOS

Es importante saber que todo elemento en python es un objeto. Y que cada objeto posee sus propios métodos (funcionalidades).

Entre los objetos más comunes que se tienen están: strings, listas y diccionarios. Cada uno de ellos posee métodos propios para ejecutar alguna funcionalidad. Por ejemplo:

cadena.count("A") #Cuenta el número de veces que se repite el carácter "A" en la cadena lista.append(1) #Agrega el entero 1 a la lista diccionario.get("cristian") #Obtiene el elemento del diccionario cuya clave es "cristian"

Veamos algunos de los métodos más importantes:

Strings formateo: capitalize()(la cual convierte la primera letra de una cadena a mayúsculas), upper()(esta función devuelve una copia de la original en mayúsculas todas las letras), lower()(esta función devuelve una copia de la original en minúsculas todas las letras), title() (convierte la cadena en formato titulo), center(spaces)(devuelve una cadena del ancho especificado, si una cadena es muy larga lo rellena con espacios), strip()(suprime blancos un otro carácter especificado del final o del principio de una expresión de serie) operaciones: count(sub-cadena)(devuelve el recuento de cuántas veces aparece un objeto dado en la lista), find(sub-cadena)(sirve para buscar patrones en sub-cadenas) replace(old, new)(reemplaza un objeto "old" por uno nuevo "new") verificación: isalnum()(comprueba si la cadena es alpha numérica), isalpha()(comprueba si la cadena es solo alfabética, que solo tiene letras), isdigit()(comprueba si la cadena solo tiene dígitos), isdecimal()(verifica si todos los caracteres de una cadena son decimales), isupper()(verifica si todos los elementos de una cadena están en mayúsculas), islower()(verifica si todos los elementos de una cadena están en minúsculas) Indexado: [índice] (muestra el índice señalado, empieza desde 0 hasta la longitud de la cadena-1) Slicing: [índice1:índice2:salto] (mira varios índices de la cadena con un salto entre ellos)

Listas operaciones: append(value)(agrega un item "value" al final de la lista), insert(index, value)(para insertar un elemento en un índice (una posición) particular de la lista), pop(index)(elimina y retorna un elemento de una lista), remove(value)(elimina valores u objetos de la lista), count(value)(devuelve el recuento de cuántas veces aparece un objeto dado en la lista) ordenado: sort()(nos permite ordenar una lista en orden ascendente o descendente), reverse()(nos permite revertir la lista) almacenamiento: clear()(borrar la pantalla de la consola) copy()(genera una copia de la lista, la cual podremos nombrar posteriormente) Indexado: [índice] Slicing: [índice1:índice2:salto]

Tuplas Operaciones: index(value)(devuelve la primera aparición / índice del elemento en la lista dada como argumento de la función), count(value)(devuelve el recuento de cuántas veces aparece un objeto dado en la tupla) Indexado: [índice] Slicing: [índice1:índice2:salto]

Diccionarios Extracción: ítem()(devuelve una lista con los keys y values del diccionario), keys()(accede a todas las claves dentro del diccionario), values()(hace una lista que muestra todos los valores dentro del diccionario), get(key)(método convencional para acceder al valor de una clave.) Eliminar: pop(key)(elimina con la key un elemento del diccionario y devuelve su valor, si no lo encuentra da error) Almacenamiento: clear()(borrar la pantalla de la consola), copy()(genera una copia del diccionario, la cual podremos nombrar posteriormente) Indexado: [key]

[ ]:
```
#  13-09-2022
#  Ejercicios métodos de strings

cadena = "MundoCruel"

"""
Buscar métodos que hagan lo siguiente:
    * Retornar caracteres en mayusculas
    * Retornar caracteres en minusculas
    * Retornar el numero de veces que se repite un caracter
    * Retornar si la cadena es alfabetica
    * Retornar si la cadena es alfanumerica
    * Retornar si la cadena contiene numeros
    * Reemplazar una letra o palabra por otra
"""

cadena.upper()      #===> Retorna una nueva cadena en mayuscula
cadena.lower()      #===> Retorna una nueva cadena en minuscula
cadena.count("M")   #===> Retorna un entero, como conteo del numero de veces que␣
 ↪se repite un string
cadena.isalpha()    #===> Retorna un booleano, como respuesta a la pregunta: ¿Es␣
 ↪alfabético?
cadena.isalnum()    #===> Retorna un booleano, como respuesta a la pregunta: ¿Es␣
 ↪alfanumérico?
cadena.isnumeric() #===> Retorna un booleano, como respuesta a la pregunta: ¿Es␣
 ↪numérico?
cadena.replace("Cruel", "Feliz")  #===> Retorna una nueva cadena, con algún␣
 ↪elemento reemplazado

# Las cadenas son inmutables, ningún método la puede alterar
# por mucho se puede generar otra cadena, pero no se afecta a la original


#-------------------------------------------------------------------------

#====== Indexación y slicing
```

```
# Indexado: operación para extraer caracteres ubicados en un indice. Ejemplo:␣
 ↪cadena[indice]
# Slicing es la operación para extraer rebanadas de una cadena. Ejemplo:␣
 ↪cadena[inicio:fin:salto]


cadena = "Mundo UnalA"

#cadena de 2 en 2
cadena[0:len(cadena):2]

#cadena al revés
cadena[-1::-1]

#elementos ubicados en las posicions 3,5 y 7
cadena[2:8:2]

#elementos hasta la mitad de la cadena
cadena[0:5]

#elementos de la mitad en adelante de la cadena
print(cadena[6:11])

#elementos desde la mitad de la cadena hasta el primer elemento
cadena[4::-1]



#----------------------Ejercicios de practica -----------------

cadena = "Anita no lava la tina de lunes a viernes"
# Extraer el primer elemento
# Extraer el ultimo elemento
# extrar los dos elementos de la mitad
# Extraer la cadena al revés
# Extraer del 10mo al 15avo elementos al revés
# Extraer cada 10mo elemento


"""
Buscar y utilizar métodos para hacer lo siguiente:
    * pasar la cadena a mayusculas
    * pasar la cadena a minusculas
    * reemplazar la palabra Anita y colocar su nombre
    * pasar la cadena a formato de titulo
    * contar el numero de veces que aparecen las vocales
```

```
"""
```

[ ]:
```
#29-09-2022

# Métodos de las listas

lista = [1, "a", 2, 3, "b"]
print(lista.append("1000"))
print(lista.append(10))
print(lista)
print(lista.pop(1))
print(lista.pop(4))
print(lista)


## --------------EJERCICIOS CLASE-------------
''' Sumar los elementos que se pueden sumar algebraicamente '''


lista2 = [1, 2, 'Cristian', 3, 'b', 'Pachon', 10]
print("---- sumar elementos algebraicos ---")
#----forma1
lista2.pop(5)
lista2.pop(4)
lista2.pop(2)
print(sum(lista2))

#---forma2
 #lista2.pop(2)
 #lista2.pop(3)
 #lista2.pop(3)
 #print(sum(lista2))

#----forma3
#for indice in [2,3,3]:
#  lista2.pop(i)
 #print(sum(lista2))

"""2) Sumar los elementos que se pueden sumar algebraicamente sin afectar lista3␣
 ↪"""
print("-------Sumar sin afectar la lista original-----")

lista3 = [1, 2, 100, 3, 'b', 'Pachon', 10, "holamundo", 5]
lista3Copia = lista3.copy()
lista3Copia.pop(4)
lista3Copia.pop(4)
lista3Copia.pop(5)
```

```
print("lista original sin afectar => ", lista3)
print("lista3 copia afectada => ", lista3Copia)
print("suma resultante =>", sum(lista3Copia))

"""3) Organice los elementos de la lista3Copia con el metodo sort. ¿Es inplace?
 ↪"""

print("-------Organizar lista 3 (copia)-----")
lista3Copia.sort()  #Por defecto ordena de forma ascendente
print(lista3Copia)




"""Metodos Indexado y Slicing"""
print("----Ejercicios indexado y slicing----------")
listaNueva = [1, 2, 3, 4, 5, "hola", "cruel", "mundo", 100]

#Indexado
#Extraer el primer elemento de 2 maneras
print(listaNueva[0], listaNueva[-9])
#Extraer el ultimo elemento de dos maneras
print(listaNueva[8], listaNueva[-1])
#Extraer el elemento de la mitad de dos maneras
print(listaNueva[4] + [listaNueva[-5]])



#Slicing
#Extraer cada elemento de dos en dos => 1, 3, 5, "Cruel", 100
print(listaNueva[0:9:2], listaNueva[0:9:2])
#Extraer hasta la mitad de la cadena => 1,2,3,4,5
print(listaNueva[0:5:1], listaNueva[0:5:], listaNueva[0:5])
#Extraer desde la mitad de la lista en adelante => 5, "hola", "cruel", "mundo",
 ↪100
print(listaNueva[4:9:1], listaNueva[4:9], listaNueva[4::])
#Extraer los elementos que son strings => "hola", "cruel", "mundo"
print(listaNueva[5:8])
#Extraer los elementos que son enteros => 1, 2, 3, 4, 5, 100
print(listaNueva[0:5] + [listaNueva[-1]])
```

[ ]:
```
# 04-10-2022


#  Diccionarios y sus métodos

# Son elemetos compuestos por clave: valor
```

```python
diccionario = {
                "Cristian Pachon": 3.0,
                "Daniel Quintero": 4.0,
                "Esteban Chavez": 5.0,
                "Margarita María": 3.0
              }

#Acceder a la calificacion de Cristian Pachon

print(diccionario["Cristian Pachon"])
print(diccionario["Margarita María"])
print(diccionario.get("Juan David Gonzalez", "no existe esta clave"))

try:
    print(diccionario["Juan David Gonzalez"])
except:
    print("Esto es un error, el programa continua")


# Extraer todas las claves del diccionario
print(diccionario.keys())

#Extraer todos los valores del diccionario
print(diccionario.values())


#Ejercicio: imprimir todas las claves de diccionario
#           utilizando un ciclo for
for key in diccionario.keys():
    print(key)

#Ejercicio: imprimir todos los valores del diccionario
#           utilizando un ciclo for
for value in diccionario.values():
    print(value)

#Ejercicio: imprimir todas las parejas clave-valor
#           de diccionario utilizando un ciclo for

for key in diccionario.keys():
    pareja = key + "-" + str(diccionario[key])
    print(pareja)    #Forma 1
    #print("{}-{}".format(key, diccionario[key])) #Forma2
    #print(key, "-", diccionario[key]) #con espacio #Forma3
```

```python
# Imprimir las parejas utilizando el método items

for pareja in diccionario.items():
    print(pareja[0] + "-" + str(pareja[1]))

for key, value in diccionario.items(): #Desempaquetado
    print(key + "-" + str(value))


#Como cambiar los valores del diccionario
# Cambiar la calificacion de cristian a 5.0
diccionario["Cristian Pachon"] = 5.0
print(diccionario)

#Ejercicio: Cambiar todos los valores del diccionario
#           a 0.0

for key in diccionario.keys():
    diccionario[key] = 0.0

print(diccionario)

#Ejercicio Cambiar los valores del diccionario de la
#          siguiente manera:
#          * Si es hombre 0.0, si es mujer 5.0


for nombre in diccionario.keys():
    if nombre[-1] == "a":
        diccionario[nombre] = 5.0
    else:
        diccionario[nombre] = 0

print(diccionario)

# Haga una copia de diccionario y cambie las calificaciones
# a 5.0

diccionarioCopia = diccionario # Esto no es una copia real
diccionarioCopia = diccionario.copy() # Esto si es una copia real

for nombre in diccionarioCopia.keys():
    diccionarioCopia[nombre] = 5.0

print(diccionario, "no se ve afectado diccionario")
print(diccionarioCopia, "solo se ve afectado diccionarioCopia")
```

```
# Cómo eliminar un par clave:valor del diccionario ¿?
del diccionario["Cristian Pachon"]
print(diccionario)


#Ejercicio: Con lo visto anteriormente,
#           como cambiar la clave Margarita Maria
#           a Margarita Rosa

valor = diccionario["Margarita María"]
del diccionario["Margarita María"]
diccionario["Margarita Rosa"] = valor
print(diccionario)

#Almacene en un diccionario la siguiente base de datos
# de calificaciones
#Almacenelos de tal manera que sea posible acceder
# a la calificacion utilizando el nombre y la materia


"""
                            Materias
    Nombre           Cuantica Etica Deportes Lenguas
Juan Gutierrez         2.0     5.0     1.3     3.2
Maria Snowden          3.1     4.9     2.2     1.1
Pedro Gonzalez         5.0     3.8     3.1     4.1
Angelica Lozano        2.1     2.7     4.1     3.2
Pablo Iglesias         3.2     1.6     5.0     1.2
Mariana Pajon          2.2     2.5     4.9     3.3
Esteban Loaiza         2.1     3.4     3.8     4.3
Daniela Pineda         5.0     4.3     2.7     1.2
Esteban Vazco          3.1     5.0     1.6     3.2
Enilse Lopez           5.0     2.2     2.5     5.0
Cristian Playonero     0.5     1.1     3.4     3.2
 """

#06-10-2022

#A pedal ===>
calificaciones = {
    "Juan Gutierrez":     {"Cuantica" : 2.0,  "Etica": 5.0,  "Deportes": 1.3, ␣
 ↪"Lenguas": 3.2},
    "Maria Snowden":      {"Cuantica" : 3.1,  "Etica": 4.9,  "Deportes": 2.2, ␣
 ↪"Lenguas": 1.1},
    "Pedro Gonzalez":     {"Cuantica" : 5.0,  "Etica": 3.8,  "Deportes": 3.1, ␣
 ↪"Lenguas": 4.1},
```

```
    "Angelica Lozano":    {"Cuantica" : 2.1,  "Etica": 2.7,  "Deportes": 4.1, ␣
 ↪"Lenguas": 3.2},
    "Pablo Iglesias":     {"Cuantica" : 3.2,  "Etica": 1.6,  "Deportes": 5.0, ␣
 ↪"Lenguas": 1.2},
    "Mariana Pajon":      {"Cuantica" : 2.2,  "Etica": 2.5,  "Deportes": 4.9, ␣
 ↪"Lenguas": 3.3},
    "Esteban Loaiza":     {"Cuantica" : 2.1,  "Etica": 3.4,  "Deportes": 3.8, ␣
 ↪"Lenguas": 4.3},
    "Daniela Pineda":     {"Cuantica" : 5.0,  "Etica": 4.3,  "Deportes": 2.7, ␣
 ↪"Lenguas": 1.2},
    "Esteban Vazco":      {"Cuantica" : 3.1,  "Etica": 5.0,  "Deportes": 1.6, ␣
 ↪"Lenguas": 3.2},
    "Enilse Lopez":       {"Cuantica" : 5.0,  "Etica": 2.2,  "Deportes": 2.5, ␣
 ↪"Lenguas": 5.0},
    "Cristian Playonero": {"Cuantica" : 0.5,  "Etica": 1.1,  "Deportes": 3.4, ␣
 ↪"Lenguas": 3.2},
}

#Automaticamente =>
data = [["Juan Gutierrez",        2.0,     5.0,     1.3,     3.2],
        ["Maria Snowden",         3.1,     4.9,     2.2,     1.1],
        ["Pedro Gonzalez",        5.0,     3.8,     3.1,     4.1],
        ["Angelica Lozano",       2.1,     2.7,     4.1,     3.2],
        ["Pablo Iglesias",        3.2,     1.6,     5.0,     1.2],
        ["Mariana Pajon",         2.2,     2.5,     4.9,     3.3],
        ["Esteban Loaiza",        2.1,     3.4,     3.8,     4.3],
        ["Daniela Pineda",        5.0,     4.3,     2.7,     1.2],
        ["Esteban Vazco",         3.1,     5.0,     1.6,     3.2],
        ["Enilse Lopez",          5.0,     2.2,     2.5,     5.0],
        ["Cristian Playonero",    0.5,     1.1,     3.4,     3.2]]

diccionarioCalificaciones = {}
for estudiante in data:
  diccionarioCalificaciones[estudiante[0]] = {"Cuantica": estudiante[1], "Etica":
 ↪estudiante[2] ,
                                              "Deportes":estudiante[3],␣
 ↪"Lenguas":estudiante[4]}
print(diccionarioCalificaciones)

"""
Calcular el promedio de calificaciones de cada estudiante usando␣
 ↪diccionarioCalificaciones
Determinar los 3 estudiantes con mejor y peor promedio
Calcular el promedio de las 4 materias
"""
```

1.5 11-10-2022

Funciones

son el elemento básico del paradigma de programación modular. sirven para evitar la repetición del código, agrupándolo en secciones, que luego podrán ser llamadas

Notación

def (parámetros...): return

como llamar a la función en python?

variableRecibida = (parámetros)

Nota: la diferencia entre la notación de una clase y una función es que la función siempre va a ir en minúsculas mientras que las clases son en mayúsculas

ejemplo:

def saludarEstudiante(nombre):

```
saludo= "hola" + " " + nombre

return saludo
```

saludoRecibido = saludarEstudiante("Alejandro")

Nota: no es lo mismo usar el print() que usar el return. el return se guarda en la memoria y el print no guarda en la memoria. aunque los dos se muestran en la pantalla. en las funciones es importante usar el return, porque si no se hace, lo que nos devuelve es un "none".

[ ]:
```
#11-10-2022

def saludarEstudiante(nombre):

    saludo= "hola" + " " + nombre

    return saludo

saludoRecibido = saludarEstudiante(" viejo sabroso")

print(saludoRecibido)

"""desarrollar una funcion que reciba dos numeros y devuelva la suma de ambos """

def suma(a,b):
    sum= a+b
    return sum

sumatoria=suma(4,5)
print(sumatoria)
```

```
"""desarrollar una funcion que reciba dos listas y devuelva una nueva lista que␣
 ↪sume
elemento por elemento"""

def sumlist(lista1,lista2):
    if len(lista1)==len(lista2):
        c=[]

        for i in range(0,len(lista1)):
            suma=lista1[i]+lista2[i]
            c.append(suma)
    else:
        print("tiene que tener el mismo tamaño las dos listas, cambie las␣
 ↪listas")
        c="none"

    return c

sumador=sumlist([1,2,3],[2,5,6])
print(sumador)


"""desarrollar una funcion que no reciba parametros y que no retorne valores,
pero que sirva para imprimir un mensaje de tres lineas"""

def cosaRara():
    print("-")
    print("-")
    print("-")


a=cosaRara()

""" desarrollar una funcion, que reciba un diccionario de calificaciones
(nombre: calificacion) y retorne el promedio del curso """

def promediator(calificaciones):
    total=sum(calificaciones.values()) # el arreglo values saca los valores de␣
 ↪ese diccionario
    prom=total/(len(calificaciones))
    return prom
promedio=promediator({"melany":2.5, "elkin":3.0 })
print(promedio)


"""desarrllar una funcion que reciba una contidad indeterminada de numeros
y retorne su producto """
```

```python
def productIndeterminado(*numeros): # * hace que se desempaquete en un numero␣
↪indeterminado de variables
    resultado=1
    for numero in numeros:
        resultado= resultado*numero
    return resultado

a=productIndeterminado(2,3,4,5)

print(a)
```

1.6 CURSOS EXTRAS

1.6.1 INTRODUCCIÓN A LAS CIENCIAS DE LA COMPUTACIÓN

Máximo común divisor Para calcular el máximo común divisor de la forma usual deberíamos descomponer los números en sus divisores primos y tomar todos los números que los compongan elevados a su máximo exponente

numero primo: es un numero natural mayor o igual a 2 que es solamente divisible por 1 y por si mismo.

algoritmo de Euclides: sean u, v dos números naturales. si u>v entonces MCD(u,v)=MCD(u-v,v) y este algoritmo tiene unas propiedades. MCD(u,v)=MCD(v,u) MCD(u,u)=u ahora lo único que tenemos que hacer es aplicar estas propiedades del máximo común divisor tomando a u y v como variables.

Tipos de datos

Booleanos: el tipo de datos booleanos denota el rango de valores lógicos que consiste de dos valores True y False y esta definido de la siguiente manera. type(booleano) = (False, True). y tienen asociadas las operaciones and (conjunción lógica), or(disyunción lógica, inclusiva) y not(negación lógica). Hay otras operaciones que producen un resultado booleano, estas son las operaciones relacionales, que son ==(igualdad relacional), /=(desigualdad relacional), <(operador menor que), <=(operador menor o igual que), >(operador mayor que) y >=(operador mayor o igual que) los últimos 4 operadores solo se pueden aplicar a datos ordenados.

leyes del álgebra de Bool: siendo p, q, r proposiciones

(p or q) or r = p or (q or r)
(p and q) and r = p and (q and r)

p or False = p p and True = p

p or q = q or p p and q = q and p

p or (q and r) = (p or q) and (p or r) p and (q or r) = (p and q) or (p and r)

p or not p = True p and not p = False

p or True = True p and False = False

not(p or q) = not p and not q not(p and q) = not p or not q

p or p = p p and p = p

not(not p)= p

not False = True not True = False

enteros: si un cierto sistema especifica el tipo entero como el conjunto de todos los números enteros con valor absoluto abs(x)<= max. siendo max el valor máximo de la extensión de la palabra, esto esta dado por la cantidad de memoria usado en la representación del numero. y si denotamos las operaciones +,-,*,/,//, % debemos hacer que el valor absoluto del resultado también sea menor que max. En Python cuando se llena la extensión de la palabra empezamos otra vez a "contar" desde el menor numero de la extensión de la palabra, digamos que da la vuelta.

ordinales: un tipo de dato tal que sus elementos definen una sucesión ordenada de elementos. del valor mas pequeño al mas grande definidos por el lenguaje. para los tipos de ordinales hay 2 funciones succ() y precc() que muestran el dato siguiente y el anterior de un cierto numero de una sucesión, respectivamente.

reales: los números reales son densos e infinitos, y el tipo real no puede representar exactamente a los números reales, ya que es un conjunto finito no ordinal, y los conjuntos finitos no pueden representar a un conjunto infinito. también están ligados a la extensión de la palabra, pasando lo mismo que lo que pasa con los enteros. sus operaciones asociadas son todas las antes vistas.

char: el tipo char denota un conjunto de caracteres, finito y ordenado. en Python tenemos que es el sistema ASCII. podemos pasar de los datos tipo entero a char(dado por su valor en ASCII) con las operaciones siguientes: ord(c) es el numero entero, llamado ordinal, del carácter en el conjunto de caracteres ASCII chr(i) es el carácter con numero ordinal i

y se tienen las siguientes relaciones

chr(ord(c)) = c ord(chr(i)) = i

1.6.2 Complejidad de un algoritmo

se mide en temporal (siendo el tiempo que se demora haciendo el programa) y de forma espacial (que es la cantidad de memoria RAM gastada al hacer el algoritmo)

1.6.3 Sucesiones

sirven para representar conjuntos de objetos donde interesa tener una noción de orden lineal entre ellos, definido en términos de la posición.en Python usamos las listas.

definición: sea A un conjunto, una sucesión finita con valores en A es una correspondencia que asocia cada número i perteneciente a {1, 2, . . . , n} con un elemento y solo un elemento de A. en Python el primer elemento es denotado por el 0 ya que se toman los índices, envés de las posiciones.

2 sucesiones son iguales si sus correspondientes elementos son iguales. si tiene diferente longitud no serán iguales.

1.6.4 Gestión memoria

generalmente Python hace una gestión de memoria, que se adapta siempre dejando disponible memoria muy superior a la necesaria para una cierta operación, mas sin embargo podemos gestionar la memoria pidiendo que guarde en unos bits en especifico. por ejemplo usando la función float8(n), estamos haciendo que el numero n este guardado en una memoria de 8 bits como un flotante (esta función es de la librería numpy), esto generalmente lo hacemos para librar al computador de complejidad espacial del algoritmo, y es perfecto para optimizar.

generalmente es mejor utilizar tuplas envés de listas para guardar información, ya que las listas son mucho más lentas en procesamiento por la cantidad de métodos que tienen, que las tuplas, las cuales que por ser invariables tienen pocos métodos, específicamente 2 .count(x)(cuenta cuantas veces está x) y .index(y)(dice en que posición está y).

una función que nos dice el tamaño que ocupa en la memoria una cierta variable es getsizeof(), que nos da el tamaño de la variable en bytes.

1.6.5 Clases

estructura que almacena funciones, nosotros podemos crearla y descargarlas, también se llama librería. una clase se llama de la siguiente forma

import nombre_clase

o también

import nombre_clase as nc

nc representa un diminutivo para reducir la escritura. y cuando necesitamos una función de esa librería, después de importarlo.

nombre_clase.función()

veremos dos librerías.

random import random

algunos arreglos o funciones de esta librería son:

.randrange(a,b) = da un numero aleatorio entre a y b .choice(lista) = selecciona un elemento de la lista, tupla o un string escogidos .random() = muestra un valor entre 0 y 1 aleatorio

math import math

algunos arreglos o funciones de esta librería son:

.factorial(n) = dá el factorial del numero n .pi = es el valor del numero pi .e = es el valor del numero e .sin(x) = saca el valor del seno de x debemos tener en cuenta que también están las funciones arco-seno (asin()), arco-coseno (acos()), coseno (cos()), tangente (tan()), arco-tangente (atan()), etc... .log(a) = saca el logaritmo natural de a .log10(b) = saca el logaritmo en base 10 de b y solo variando el numero del logaritmo lo tenemos en varias bases .pow(c,d) = eleva a la base c a la potencia d .sqrt(p) = saca la raíz cuadrada del numero p .degrees(q) = pasa el numero q a grados

[ ]:
```
# maximo comun divisor
n=int(input("por favor dé el primer numero"))
m=int(input("por favor dé el segundo numero"))

while n != m :
    if a>b:
        a = a-b
    else:
        b = b-a
print("el MCD de los numeros es: ", a)

# gestion de memoria
import numpy as np
a = np.int8(4)
print(a.nbytes)
b = np.int64(4)
print(b.nbytes)
c = np.float32(4)
print(c.nbytes)
d = np.float64(4)
print(d.nbytes)
e=np.array([2,2,5]).astype('int32')
print(e.nbytes)
f=np.array("H")
print(f.nbytes)
g=np.array("HOLA")
print(g.nbytes)

# libreria random
import random
print('imprime un numero aleatorio entero=',random.randrange(3,10))
print('selecciona aleatoriamente=',random.choice(["uno", "dos", "tres"]))
print('imprime un numero aleatorio decimal entre 0 y 1=',random.random())
print('imprime una letra aleatoria=',random.choice("AEIOU"))

# libreria math
import math
print("Factorial=",math.factorial(3))
print("PI=",math.pi)
print("E=",math.e)
print("seno=",math.sin(math.pi/6))
print("aseno=",math.degrees(math.asin(0.5)))
print("Logaritmo Natural=",math.log(math.exp(2)))
print("Logaritmo en base 10=",math.log10(100))
print("Logaritmo en base 2=",math.log2(8))
print("Potencia=",math.pow(3,2))
print("Raiz cuadrada=",math.sqrt(4))
```