RELATORIA CLASE DE INFORMATICA III

ARELIS BARON GOMEZ

UNIVERSIDAD NACIONAL DE COLOMBIA

INFORMATICA III

CRISTIAN ELIAS PACHON PACHECO

26 DE MARZO DE 2023

CLASE No 1

Fecha: 17/02/2023

Tema: Introducción al curso Informática III, entorno Python

Objetivo: crear y actualizar un repositorio local y enlazarlo a la nube.

RESUMEN

En esta clase se creó y enlazo a la nube un repositorio local, para esto se debe crear una carpeta en el sistema de archivos y abrirla desde Visual Studio Code (VSC). Luego, se debe ir al icono del código fuente y dar clic en *"Initialize repository".*

Para actualizar el repositorio, se debe ir al icono del código fuente y hacer clic en el icono "+" para agregar los cambios realizados en la "stage area". Después, se debe hacer clic en el botón "Commit" y dejar un mensaje antes de hacer el commit.

Para enlazar el repositorio a la nube, primero se deben configurar el usuario y el correo electrónico en la terminal con los comandos *"git config --global user.name" y "git config --global user.email".* Luego, se debe ir al icono del código fuente y hacer clic en el botón "Publish Branch" o "Sync Changes". Se debe esperar a que el repositorio se suba a la nube y luego verificarlo en el buscador de Google.

Palabras clave: repositorio, enlace a la nube, Visual Studio Code.

Clase 2

Fecha 24/02/2023

Tema: Tipos de Datos y Operadores

Objetivo: Estudiar los tipos de datos y operadores utilizados en Python.

RESUMEN

Este repositorio proporciona una descripción general de los tipos de datos y operadores utilizados en Python. Se describen los tipos de datos, como Strings, enteros, flotantes, booleanos, listas, tuplas, diccionarios y conjuntos, así como los operadores aritméticos, lógicos, comparativos y de pertenencia.

Contenido

TIPOS DE DATOS:

- **Strings:** son conjuntos ordenados de caracteres
  EJEMPLO: "" " "Hola mundo" '123456'
- **Booleanos:** representan valores de lógica binaria.
  EJEMPLO: True False
- **Enteros:** Son números enteros, dado que dicho conjunto es infinito, en Python el conjunto está limitado por la capacidad de la memoria disponible.
  EJEMPLO: -99999999 0 10000000
- **Flotantes:** son números con decimales.
  EJEMPLO: 100.09 -36.99 0.0

- **Listas:** son listados de datos que tienen un orden y pueden ser modificados.
  EJEMPLO: [] [0, 1, 2] ["uno", "dos", "tres"]
- **Tuplas**: son secuencias de elementos y no pueden ser modificadas directamente.
  EJEMPLO: () (0, 1, 1) ("uno", "dos", "tres")
- **Conjuntos:** son objetos que almacenan datos y no admiten elementos duplicados.
  EJEMPLO: {} {"A", "B", "C", "D"}
- **Diccionarios:** son estructuras que contienen una colección de elementos clave-valor y las claves deben ser únicas.
  EJEMPLO: {} {"hola": "hello", "mundo": "world"}

Ejemplo en Python

```
'''
1) Imprimir en la terminal
    dos enteros en una misma linea
    dos flotantes en una misma linea
    dos strings en una misma linea
    dos booleanos en una misma linea
    dos listas en una misma linea
    dos tuplas en una misma linea
    un diccionario en una misma linea
    un conjunto en una misma linea
'''

print(2,3)
print(2.1, 3.0)
print("asa", "hola")
print(True, True)
print([], [0])
print((0,),(0,1))
print({1:"uno", 2:"dos"})
print({"A", "B", "C", "D"})
```

OPERADORES ARITMETRICOS

- **Asignación:**

a = 3

b = "hola"

c = [1,2,3]

- **Aritméticos:**

Operador	Significado
+	Suma
-	Resta
*	Multiplicación
/	División
//	División Entera,
%	Residuo De Una División

- **Logicos:**

True and True => True

False or False => False

not True ==> False

- **Comparativos:**

Operador	Significado
>	mayor que
<	menor que
==	igual que
>=	mayor igual
<=	menor igual
!=	diferente de

- **Pertenencia:**

Se emplean para identificar pertenencia en alguna secuencia (listas, strings, tuplas).

in: devuelve True si el valor especificado se encuentra en la secuencia. En caso contrario devuelve False.

not in devuelve True si el valor especificado no se encuentra en la secuencia. En caso contrario devuelve False.

Ejemplo en python

```
"2) Resolver con los operadores que considere conveniente"

""" Dadas las coordenadas P1(5,4,5) y P2(0,10,9).
Realice un codigo que determine la distancia entre ambos puntos """

P1 = (5,4,5)
P2 = (0,10,9)

distancia = ((P1[0] - P2[0])**2 + (P1[1] - P2[1])**2 + (P1[2] - P2[2])**2) ** 0.5
print("distancia => ", distancia)
```

Palabras claves: Tipos de datos, strings, enteros, flotantes, booleanos, listas, tuplas, diccionarios, conjuntos, operadores.

Clase 3

Fecha: 01/03/2023

Tema: Funciones integradas de Python.

Objetivo: Estudiar las funciones integradas en Python y su uso básico.

RESUMEN

Las funciones integradas de Python son aquellas que están disponibles en el lenguaje de programación sin necesidad de importar ningún módulo adicional. A continuación se presentan algunas de las funciones integradas más utilizadas, clasificadas según su funcionalidad, junto con una breve descripción y ejemplos de su uso.

Funciones de entrada y salida:

Función	Descripción	Ejemplo

input()	Permite obtener el texto escrito por el usuario, el cual se asignará aun espacio de memoria con el nombre que el programador vea conveniente.	nombre = input("Introduce tu nombre: ")
print()	puede imprimir en pantalla varias expresiones separadas por coma o a través de las cadenas	print("Hola, mundo!")

Funciones de ayuda:

Función	Descripción	Ejemplo
help(),	muestra información de ayuda sobre un objeto, función o módulo.	help(str)
dir()	Permite devolver el conjunto de nombres asociados al objeto incluido como argumento. Si no se incluye ningún argumento, devuelve elconjunto de nombres del ámbito local.	dir(str)

Conversiones:

Función	Descripción	Ejemplo
float()	Devuelve un número real a partir de un número o de una cadena de texto.	precio = float("10.5")
list():	convierte un objeto iterable en una lista.	lista = list((1,2,3))
tuple():	convierte un objeto iterable en una tupla.	tupla = tuple([1,2,3])
set()	Devuelve la diferencia del conjunto con el iterable como un conjuntonuevo.	conjunto = set("Hola")
dict():	crea un diccionario a partir de una secuencia de pares clave-valor.	diccionario = dict([('clave1', 'valor1'), ('clave2', 'valor2')])
complex()	devuelve un número imaginario a partir de los valores cedidos comoargumentos para la parte real e imaginaria del mismo.	numero = complex(3, 4)
hex():	onvierte un número entero en su representación hexadecimal.	hexadecimal = hex(255)
oct():	convierte un número entero en su representación octal	octal = oct(8)
bin():	convierte un número entero en su representación binaria.	binario = bin(10)
int()	devuelve un número entero a partir de un número o de un string, odevuelve el valor 0 si no se incluye ningún argumento.	edad = int("25")

Secuencia (listas y tuplas):

Función	Descripción	Ejemplo
range()	Genera una secuencia de números enteros que podemos utilizar para iterar en un bucle.	numeros = range(0, 10, 2)
enumerate()	toma una secuencia o un iterador y retorna objeto de tipo enumerate.	lista = ["a", "b", "c"] for indice, valor in enumerate(lista): print(indice, valor)
zip()	toma como argumentos un número arbitrario de iteradores, y los agrega en un objeto de tipo zip.	lista1 = [1, 2, 3] lista2 = ["a", "b", "c"] for tupla in zip(lista1, lista2): print(tupla)

Operaciones con secuencias:

Función	Descripción	Ejemplo
len()	retorna el número de elementos que contiene un objeto.	len([1,2,3])
sum()	retorna el total de sumar los elementos de la secuencia iterable.	sum([1,2,3])
min()	retorna el elemento más pequeño del objeto iterable. También se pueden utilizar dos o más argumentos, en cuyo caso retorna el menor de los argumentos.	min([5,1,3,2])
sorted()	retorna una lista con los elementos de iterable ordenados de menor a mayor.	sorted([3,1,2])
map()	aplica funcion a cada uno de los elementos del objeto iterable y retorna el resultado en un objeto map.	list(map(lambda x: x**2, [1,2,3]))
filter()	extrae todos los elementos de una secuencia iterable para los cualesla funcion retorna True.	list(filter(lambda x: x > 2, [1,2,3]))
max():	Devuelve el valor máximo de una secuencia	max([5,1,3,2])
round():	Se utiliza para redondear un número con una precisión determinada	round(3.14159, 2

Palabras clave: funciones integradas, conversiones, secuencias, operaciones.

Clase 4

Fecha: 3/03/2023

Tema: Métodos de Python

Objetivo: estudiar los diferentes métodos que se pueden utilizar en Python para manipular cadenas de caracteres.

Resumen

Python es un lenguaje orientado a objetos en el que todos los tipos de datos son objetos. Cada objeto tiene sus propios métodos, que son funciones que pueden ser llamadas en ese objeto. Los métodos de los strings son especialmente útiles para manipular cadenas de caracteres.

METODOS DE STRING

método	Descripción	Ejemplo
capitalize	Retorna la cadena con la primera letra en mayúscula	cadena.capitalize()
upper	Retorna la cadena en mayúscula	cadena.upper()
lower	Retorna la cadena en minúscula	cadena.lower()
title	Retorna la cadena con la primera letra de cada palabra en mayúscula	cadena.title()
center	Retorna la cadena centrada en una cantidad específica de caracteres	cadena.center(20)
strip	Retorna la cadena sin espacios en blanco al inicio y al final	cadena.strip()

MÉTODOS DE OPERACIONES

método	Descripción	Ejemplo
count	Retorna el número de veces que aparece una subcadena en la cadena	cadena.count('o')
replace	Retorna una cadena con una subcadena reemplazada por otra	cadena.replace('mundo', 'tierra')
find	Retorna la posición de la primera aparición de una subcadena	cadena.find('mu')

MÉTODOS DE VERIFICACIÓN

método	Descripción	Ejemplo
isalpha	Retorna True si la cadena contiene sólo letras	my_string = "abc" print(my_string.isalpha())
isalnum	Retorna True si la cadena contiene sólo letras y números	my_string = "abc123" print(my_string.isalnum())
isdigit	Retorna True si la cadena contiene sólo números	my_string = "1234" print(my_string.isdigit())

Los métodos de indexing permiten acceder a un carácter en particular dentro de la cadena mediante su índice, mientras que los métodos de slicing permiten seleccionar una parte de la cadena en base a un rango de índices.

Palabras clave: métodos, cadenas de caracteres, operaciones, verificación, indexing, slicing.

Clase 5

Fecha: 8/03/2023

Tema: Métodos de las listas en Python.

Objetivo: Conocer los diferentes métodos que se pueden utilizar en las listas de Python.

RESUMEN

En esta clase se estudiaron las listas, las cuales son un tipo de dato muy útil en Python, y es importante conocer los diferentes métodos que se pueden utilizar con ellas para manipular su contenido de forma eficiente.

A continuación se presenta un cuadro comparativo con los principales métodos de las listas:

OPERACIONES

Método	Descripción	Ejemplo
append(value)	Agrega un elemento al final de la lista	lista.append(10)
insert(index, value)	Agrega un elemento en la posición especificada	lista.insert(2, "hola")
remove(value)	Elimina el primer elemento de la lista que coincide con el valor dado	lista.remove(5)
pop(index)	Elimina el elemento en la posición especificada y lo devuelve	lista.pop(2)
count()	Devuelve el número de veces que aparece el elemento en la lista	lista.count(3)

ORDENADO

Método	Descripción	Ejemplo
sort()	Ordena los elementos de la lista en orden ascendente	lista.sort()
reverse()	Invierte el orden de los elementos de la lista	lista.reverse()

ALMACENAMIENTO

Método	Descripción	Ejemplo
clear()	Elimina todos los elementos de la lista	lista.clear()
copy()	Devuelve una copia de la lista	lista_copia = lista.copy()

INDEXADO

Método	Descripción	Ejemplo

[index]	Devuelve el elemento en la posición especificada	lista[2]

SLICING

Método	Descripción	Ejemplo
[init:end:step]	Devuelve una porción de la lista desde la posición inicial hasta la posición final especificadas, con un salto indicado	lista[1:4:2]

Palabras clave: listas, métodos, almacenamiento, indexado, slicing.

Clase 6

Fecha: 10/03/2023

Tema: Métodos de los diccionarios

Objetivo: Estudiar el funcionamiento de los métodos para los diccionarios en Python.

Resumen

Un diccionario es una estructura de datos en Python que almacena pares de clave-valor. Los diccionarios tienen una serie de métodos incorporados que permiten operaciones de extracción, eliminación, almacenamiento e indexación.

A continuación, se presenta un cuadro comparativo con una breve descripción y un pequeño ejemplo en Python para cada método:

EXTRACCION

Método	Descripción	Ejemplo
keys()	Devuelve una lista con todas las claves del diccionario	diccionario.keys()
values()	Devuelve una lista con todos los valores del diccionario	diccionario.values()
get(<clave>)	Devuelve el valor asociado a la clave especificada. Si la clave no existe, devuelve un valor predeterminado	diccionario.get("hola", "No existe")

ELIMINAR

Método	Descripción	Ejemplo
pop(<clave>)	Elimina el elemento con la clave especificada y devuelve su valor. Si la clave no existe, devuelve un valor predeterminado	diccionario.pop("mundo", "No existe")

ALMACENAMIENTO

Método	Descripción	Ejemplo

clear()	Elimina todos los elementos del diccionario	diccionario.clear()
copy()	Devuelve una copia superficial del diccionario	diccionario_copia = diccionario.copy()

INDEXADO

Método	**Descripción**	**Ejemplo**
diccionario[<clave>]	Devuelve el valor asociado a la clave especificada. Si la clave no existe, devuelve un error	diccionario["profesor"]

Palabras clave: diccionarios, claves, valores, métodos, extracción, eliminación, almacenamiento, indexación.

CLASE N 7

Fecha: 15/03/2023

Tema: Condicional if

Objetivo: Estudiar y aplicar el condicional if

RESUMEN

El condicional if permite ejecutar un bloque de código si la condición dada es verdadera. En caso de ser falsa, se ejecutan otras sentencias en la instrucción else. Además, se pueden agregar múltiples condiciones utilizando la sintaxis elif. La estructura del condicional if es la siguiente:

if <condición>:
<sentencias>
else:
<sentencias>

Si se quieren agregar múltiples condiciones, se puede utilizar la sintaxis elif:

if <condición>:
<sentencias>
elif <condición>:
<sentencias>
else:
<sentencias>

Palabras clave: condicional, if, sintaxis, elif, sentencias.

CLASE N 7

Fecha: 17/03/2023

Tema: Ciclo While y Ciclo For

Objetivo: Comprender la sintaxis y funcionamiento de los ciclos While y For en Python, así como identificar las diferencias en su uso.

RESUMEN

En esta clase se explicó el uso de los ciclos While y For en Python, el cual evalúa una condición, y en caso de ser verdadera ejecuta las sentencias contenidas en el ciclo.

Durante el ciclo la condición podría cambiar a falso, lo que ocasionaría que el ciclo termine en la siguiente ejecución.

EJEMPLO

```
Ejemplo1: Ejecutar un ciclo while infinito
          cada vez que se repita el ciclo imprimir
          un mensaje informando la repetición

condicion = True
while condicion:
    print("Se ha repetido el ciclo")

#nota: Siempre evitar ciclos infinitos
```

CICLO FOR

Se utiliza cuando se conoce el número de iteraciones o la secuencia que se debe recorrer recorrerse.

SINTAXIS

for <iterador> in <iterable>:

<sentencias>

EJEMPLO

```
Ejemplo2 : Recorrer la secuencia (1,3,5...2001) y luego contar
           cuantas veces aparece un numero multiplo de 5 y 11 a la vez

secuencia = range(1, 2002, 2)
cont = 0

for numero in secuencia:
    if (numero % 55) == 0:
        cont += 1
```

Palabras clave: Ciclo While, Ciclo For, sintaxis, iteraciones.

CLASE N 8

Fecha: 22/03/2023

Tema: Funciones

Objetivo: Conocer y estudiar las funciones en Python.

RESUMEN

En esta clase se abordó el tema de las funciones en Python, su sintaxis y su importancia como solución al paradigma de programación funcional. Además, se revisó un ejemplo de cómo crear una función que determine si un número es par o impar.

Funciones

son bloques de código que se crean para evitar la repetición de código y para organizar mejor el programa en secciones más pequeñas y manejables. La sintaxis para definir una función es a través de la palabra clave "def" seguida del nombre de la función y los parámetros que recibe. Dentro de la función se escriben las sentencias que se ejecutan y, opcionalmente, se puede retornar un valor con la palabra clave "return".

EJEMPLO

```
EJEMPLO =>  Crear una función que determine si un numero es par:

def esUnNumeroPar(numero):          # Esto es crear o definir una funcion
    esPar = (numero % 2 == 0)
    return esPar

print(esUnNumeroPar(9))             # Esto es ejecutar o llamar la función
print(esUnNumeroPar(0))             # Esto es ejecutar o llamar la función
print(esUnNumeroPar(1))             # Esto es ejecutar o llamar la función
print(esUnNumeroPar(2))             # Esto es ejecutar o llamar la función
```

Palabras clave: funciones, sintaxis, parámetros, retornar, ejecutar, resultado.

REFERENCIAS

- Torres, A. (2023, marzo 20). Sentencia If Else de Python: Explicación de las sentencias condicionales. freecodecamp.org. https://www.freecodecamp.org/espanol/news/sentencia-if-else-de-python-explicacion-de-las-sentencias-condiciones/

- Brugués, A. (2023, marzo 26). Funciones integradas en Python. Programa en Python; Albert Brugués. https://www.programaenpython.com/miscelanea/funciones-integradas/

- Tipos de datos básicos de Python. (2023, marzo 26). J2LOGO. https://j2logo.com/python/tutorial/tipos-de-datos-basicos-de-python/