RELATORIA CLASE DE INFORMATICA III

ARELIS BARON GOMEZ

UNIVERSIDAD NACIONAL DE COLOMBIA
INFORMATICA III
CRISTIAN ELIAS PACHON PACHECO
17 DE SEPTIEMBRE DE 2022

CLASE 1

Fecha: 24 de agosto 2022

Tema: Introducción al curso Informática III, entorno Python

Objetivo: Conocer el entorno en el cual se va a trabajar durante el curso.

RESUMEN

Se realizó la instalación de los programas que se van a utilizar durante el curso, se configuró el entorno de desarrollo para *Python*, por lo cual se instaló *Visual Studio Code*, Para este curso se va a implementar la herramienta de *GitHub* en donde se estipuló subir un repositorio con las clases del curso, el cual estará compuesto por los archivos de texto y practicas con los códigos.

Posteriormente Se configuró la cuenta de GitHub en visual code, se creó un usuario y contraseña en el terminal siguiendo los siguientes pasos:

1. Ir a git config, global user.name “abaron”
2. git config --global user.email "abaron@unal.edu.co"

Luego, se procedió a verificar la instalación de GitHub con git –versión. Se seleccionó la pestaña de git (los 3 nodos) y se inicializo el repositorio (working directory), se dio click al boton initialize_repository y se Llevo el repositorio al stage area : seleccionar +. Para llevar el repositorio al commit area escribimos mensaje y damos click en el boton commit, finalmente se subió el repositorio a la nube, seleccionando click en el icono de la nube.

CLASE 2

Fecha: 26 de agosto de 2022

Tema: Tipos de datos en Python

Objetivo: Estudiar los tipos de datos utilizados en Python.

RESUMEN

En esta clase se definieron los diferentes tipos de datos utilizados en Python, los cuales son un conjunto de datos que tienen propiedades y características ya establecidas, se realizaron algunos ejemplos de los mismos para interiorizar mejor la información, los datos vistos se presentan a continuación:

- **BOOLEANOS:** Son aquellos que pueden representar valores de lógica binaria, esto es 2 valores, que normalmente representan falso o verdadero.

Ejemplo: True, False

- **STRINGS:** Son un conjunto ordenado de letras (caracteres).

Ejemplo: "", '', "casa", "123", """", "12.23132"

- **ENTEROS:** Este tipo de dato comprende el conjunto de todos los números enteros, pero dado que dicho conjunto es infinito, en Python el conjunto está limitado realmente por la capacidad de la memoria disponible. No hay un límite de representación impuesto por el lenguaje**.**

Ejemplo: -1000, 1000, 0,

- **FLOTANTES**: El formato de dato del tipo “coma flotante” o “float” se aplica a los números con decimales. Los números de coma flotante tienen una mayor resolución que los de 32 bits.

Ejemplo: 12.5, -100.0, 5.

Otro tipo de datos:

- **LISTAS:** Son listados de datos que tienen un orden, por lo cual se tiene en cuenta la posición del elemento (el primer elemento es el cero, no el numero 1)

Nota: En las listas se puede modificar los elementos.

Ejemplos:

[]

[1,2,3]

["A", "B", "C"]

[1, "A", 2, "B", 3, "C"]

- **TUPLAS**: son secuencias de elementos, la diferencia con las listas es que las tuplas no pueden ser modificadas directamente (inmutable)

Nota: Para definir una tupla los elementos se separan con comas y se encierran entre paréntesis

Ejemplos:

(), (1,2,3), ("A", "B", "C"), (1, "A", 2, "B", 3, "C")

- **CONJUNTOS:** representan un objeto capaz de almacenar datos, que están indexados, son de gran utilidad para buscar elementos.

Nota: Los conjuntos no admiten elementos duplicados

Ejemplo:

{"A", "B", "C"}

- **DICCIONARIOS:** son estructuras que contienen una colección de elementos de la forma *Clave: valor* separados con comas y encerrados entre llaves, se debe tener en cuenta que las claves deben ser inmutables y los valores pueden ser de cualquier tipo de dato.

Nota: Las claves deben ser únicas en los diccionarios.

Ejemplo:

{"CRISTIAN": 5, "DANIELA": 0, "SEBASTIAN": 3}

CLASE 3

Fecha: 26 de agosto de 2022 y 31 de agosto de 2022

Tema: Operadores

Objetivo: Estudiar los diferentes operadores en Python.

Resumen: En esta clase se estudiaron los diferentes operadores que se utilizan en Python, los cuales le indican al interprete que realice una operación específica, como aritmética, comparación, lógica, etc. Estos son los diferentes tipos de operadores en Python:

BOOLEANOS:

Operación	Resultado
X or y	si *x* es falso, entonces *y*, si no, *x*
X and y	si *x* es falso, entonces *x*, si no, *y*
Not x	si *x* es falso, entonces True, si no, False

Ejemplo:

```
print(1 and 1)
print(1 and 0)
print(19 and 20)
print(0 and 100)
print(100 or 0)
print(-2 or 20)
print("cristan" and "elias")
print("Unal" and " ")
```

ENTEROS Y FLOTANTES : Operaciones aritméticas

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

Ejemplo:

```
print("-------------Operadores aritméticos------------")
print("suma            ==>", 1+2+3)
print("mult y suma     ==>", 1*2+3)
print("suma y mult     ==>", 1+2*3)
print("division entera ==>", 5//3)
print("division entera ==>", 0//3)
print("residuo         ==>", 5%2)
print("residuo         ==>", 13%9)
print("potenciacion    ==>", 9**2)
print("potenciacion    ==>", 9**0.5)
```

OPERACIONES DE COMPARACION

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

Ejemplo:

```
print("-------Operadores de comparacion----")
print("mayor         ==> ", 1 > 2)
print("menor         ==>", 19 < 0)
print("menor o igual ==>", -1001 <= -1001)
print("igual         ==>", 3 == -5)
print("mayor o igual ==>", 19 >= 20)
print("diferente de  ==>", 30 != 31)
```

OPERADORES TIPO STRINGS:

+, concatenación o unión de STRINGS: Si se concatena dos o más strings con el operador + da como resultado un nuevo string . Como se puede observar, para concatenar dos cadenas con el operador + simplemente se requieren dos objetos de tipo string.

Ejemplo:

```
print("concatenacion ==>", "hola" + " mundo")
print("replicacion   ==>", "a" * 10)
print("concatenacion ==>", [1] + [1,2,3])
print("repiclacion   ==>", [0] * 10)
```

OPERADORES ENTRE CONJUNTOS:

Operador	Significado
|	Unión

&	intersección
-	Diferencia

OPERADORES DE PERTENENCIA:

Se emplea 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.

CLASE 4

Fecha: 02 de septiembre de 2022

Tema: Funciones Integradas de Python.

Objetivo: Estudiar las diferentes funciones integradas de Python.

RESUMEN

En esta clase se vieron funciones integradas de Python, estas se pueden utilizar sin necesidad de importar ningún modulo, a la hora de programar en Python, es posible que suenen palabras como file(), print(), range(), etc. A esto se le llaman funciones integradas. Es decir, funciones que nos proporciona el propio lenguaje que se pueden ejecutar llamadas.

Las funciones se componen de un conjunto de instrucciones combinadas para obtener algún resultado y son ejecutadas mediante llamadas a la misma función. Los resultados en Python pueden ser o bien la salida de algún cálculo en la función Esas funciones pueden ser funciones

integradas (mencionadas anteriormente) o funciones definidas por el usuario. A continuación se describen algunas de ellas:

FUNCIONES DE ENTRADA Y SALIDA

Función	Descripción
input()	Permite obtener el texto escrito por el usuario, el cual se asignará a un espacio de memoria con el nombre que el programador vea conveniente.
print()	puede imprimir en pantalla varias expresiones separadas por comas o a través de las cadenas
format()	Permite múltiples sustituciones y formateo de valores. Este método nos permite concatenar elementos dentro de una cadena a través del formato posicional.

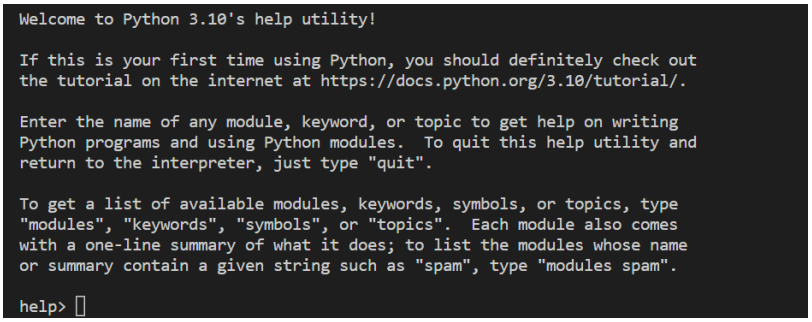
Ejemplo:

```
clave_real = "Unal2022"
clave_usuario = input("Ingrese su clave: ")
print((clave_usuario == clave_real and "La clave es correcta") or "La clave es incorrecta")
```

FUNCIONES DE AYUDA

Función	Descripción
help(),	Permite invocar el sistema de ayuda integrado de Python. Si no se añade ningún argumento, la función devuelve un mensaje de bienvenida explicando el uso de la función e invitándonos a introducir el texto a buscar.
dir()	Permite devolver el conjunto de nombres asociados al objeto incluido como argumento. Si no se incluye ningún argumento, devuelve el conjunto de nombres del ámbito local.
type()	**El método devuelve el tipo de clase del argumento(objeto) pasado como parámetro. Se utiliza con fines de depuración**

Ejemplo:



FUNCIONES MATEMATICAS:

abs()	devuelve el valor absoluto de un número.
round()	redondea los números hasta el valor entero o decimal más cercano.
pow()	devuelve el resultado de elevar x a y.

Ejemplo:

```
3   abs(3)
4   abs(-3)
5   abs(-3.0)
6   abs(3+4j)
7
```

FUNCIONES DE CONVERSIONES:

Función	Descripción
int()	devuelve un número entero a partir de un número o de un string, o devuelve el valor 0 si no se incluye ningún argumento.
float()	devuelve un número real a partir de un número o de una cadena de texto.
str()	Tipo de variable para guardar texto.
complex()	devuelve un número imaginario a partir de los valores cedidos como argumentos para la parte real e imaginaria del mismo.
bool(),	convierte un valor x en su equivalente booleano.
set()	Devuelve la diferencia del conjunto con el iterable como un conjunto nuevo.

list()	No es una función en sí, se trata de un constructor para listas. Cuando se utiliza sin argumentos crea una lista vacía. También puede pasar como parámetro una secuencia iterable, en cuyo caso la convierte a una lista. Normalmente esa secuencia iterable que pasa a list() no suele ser una lista ya que estaría generando código redundante.
tuple()	Es el constructor de la clase tupla. Se puede usar sin argumentos para crear una tupla vacía, o pasarle un objeto iterable cuyos elementos serán los elementos de la tupla. En este sentido, se puede utilizar *tuple()* para convertir un *string* o una lista a una tupla.
bin()	Convierte un número entero en su representación binaria con formato de texto.
oct()	Toma un número entero y devuelve su representación octal en formato de cadena.
hex()	Se utiliza para convertir un número entero en su forma hexadecimal correspondiente.
int()	Devuelve un número entero a partir de un número o de un *string,* o devuelve el valor 0 si no se incluye ningún argumento

GENERACIÓN DE SECUENCIAS ITERABLES:

Función	**Descripción**
range()	Genera una secuencia de números enteros que podemos utilizar para iterar en un bucle.
enumerate()	toma una secuencia o un iterador y retorna objeto de tipo enumerate.
zip()	toma como argumentos un número arbitrario de iteradores, y los agrega en un objeto de tipo zip.

OPERACIONES CON SECUENCIAS:

Función	**Descripción**
len()	retorna el número de elementos que contiene un objeto.
sum()	retorna el total de sumar los elementos de la secuencia iterable.
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.
sorted()	retorna una lista con los elementos de iterable ordenados de menor a mayor.
map()	aplica funcion a cada uno de los elementos del objeto iterable y retorna el resultado en un objeto map.

filter()	extrae todos los elementos de una secuencia iterable para los cuales la funcion retorna True.

CLASE 5

Fecha: 07 de septiembre de 2022

Tema: Condicional If

Objetivo: Estudiar y aplicar el condicional if

RESUMEN

En esta clase se trabajó con el Condicional if el cual permite ejecutar un conjunto de sentencias, en caso de que una condición sea verdadera. En caso de que la condición sea falsa, las sentencias contenidas son ignoradas.

Para implementar el if en un código se debe implementar la siguiente notación:

if <condición>:

<sentencia1>

<sentencia2>

<sentencia3>

.....

Ejemplo:

```python
## determine si un numero es mayor a 18, utilizando el condicional if

numero = 18
if numero > 18:
    print("Numero es mayor a 18")
```

Cuando se requieren mas opciones se utiliza el **elif**, es decir, "Si la primera condición es verdadera, realiza esto, si no, realiza esto otro", con esta sentencia le indicamos al programa, "Si esto no es verdadero, intenta esto otro, y si todas las condiciones fallan en ser verdaderas, entonces haz esto.

Ejemplo:

```python
## Pedir a el usuario que ingrese 3 numeros, luego imprimir el numero mayor y el menor

numero1 = int(input("Ingrese primer numero: "))
numero2 = int(input("Ingrese segundo numero: "))
numero3 = int(input("Ingrese tercer numero: "))

mayor = 0

if (numero1 >= numero2) and (numero1 >= numero3):
    mayor = numero1
elif (numero2 >= numero1) and (numero2 >= numero3):
    mayor = numero2
elif (numero3 >= numero1) and (numero3 >= numero2):
    mayor = numero3

print("El numero mayor es {}".format(mayor))
```

CLASE 6

Fecha: 09 de septiembre de 2022

Tema: Ciclo While

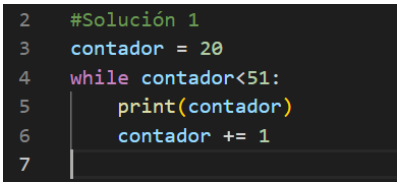
Objetivo: Estudiar y aplicar mediante ejemplos el ciclo While

RESUMEN

En esta clase se estudio el ciclo while 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:



Este código imprime los números del 20 al 50 con el ciclo while

REFERENCIAS

- Torres, A. (2021, septiembre 22). *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. (2021, septiembre 26). *Funciones integradas en Python*. Programa en Python; Albert Brugués. https://www.programaenpython.com/miscelanea/funciones-integradas/
- Tipos de datos básicos de Python. (2020, marzo 6). J2LOGO. https://j2logo.com/python/tutorial/tipos-de-datos-basicos-de-python/