Informe I

Informática III

Juan José Galindo Márquez

Universidad Nacional de Colombia, Sede Manizales
Manizales, Colombia
2022-2

Objetivos

1. Realizar un recuento detallado y de fácil acceso de la información suministrada en las clases vistas de la materia de Informática III hasta el 7 de Septiembre.

2. Aprender los fundamentos básicos del lenguaje de programación Python, desde la descarga e incorporación del lenguaje en un PC a través de un IDE hasta el uso de ciclos básicos While para la realización de acciones repetidas, condicionadas y secuenciadas.

3. Reforzar y profundizar sobre los temas básicos vistos en clase.

Clase 1

Agosto 24 del 2022

Entorno Python

En la primera clase configuramos el entorno Python donde se trabajará la materia durante el semestre 2022-2 con las siguientes herramientas:

1. Descargar e instalar el lenguaje de programación Python.
2. Descargar e instalar el editor de código Visual Studio Code.
3. Descargar e instalar el software de control de versiones GIT.
4. Crear una cuenta y usuario en la plataforma web GitHub.

Con estas herramientas vamos a crear nuestro primer repositorio en la nube al que se le llamará “CLASES”. A continuación veremos cómo configurar nuestra cuenta GitHub a Visual Studio Code.

1. Configurar el correo y usuario de GitHub en la terminal de Visual Studio Code.
   git config --global user.name "usuario"
   git config --global user.email "correo_@unal.edu.co"

2. Verificar que GIT esté instalado.
   git --version

3. Seleccionar en la pestaña a la izquierda en nuestro editor de código (3 nodos).

4. Inicializar repositorio (working directory).

5. Llevamos nuestro repositorio al stage area: seleccionar +.

6. Llevar el repositorio al commit area: escribimos mensaje con el que subiremos el repositorio y damos click en el botón commit.

7. Por último subiremos el repositorio a la nube.

Clase 2

26 de agosto del 2022

Tipos de datos en Python/Operaciones

Existen diferentes tipos de datos en Python como los siguientes:

Enteros: Este representa números enteros como:

-5, 101, 999, 6789.

Flotantes: Estos se representan como números decimales:

3,14, -69,1.

Strings: Son cadenas de caracteres alfanuméricos:

“Hola mundo pachito”, “pizza”.

Booleanos: Estos representan valores lógicos:

True, False.

Listas: Son arreglos que guardan cualquier tipo de dato (estas se guardan en [ ]):

[1, 9, 8, 5, 6, 2], [“maria”, “juan”, 69, “hola”].

Tuplas: Arreglos de carácter inmutable:

colores=[“red”, “blue”, “green”]

Diccionarios: Los diccionarios en Python son un tipo de dato que permite guardar pares ordenados de información a través de claves (no pueden existir

dos elementos con una misma clave), acompañados de un almacenamiento de información más detallado a través de las mismas con valores de cualquier tipo:

Diccionarios: {clave1:valor1, clave2:valor, clave3: valor}.

Operaciones en Python

Python nos permite realizar operaciones matemáticas con los siguientes operadores:

Operaciones con Booleanas:

Adición: +
Condición de Veracidad: True, False

Ejemplo: x = True | y = False.

Operaciones con Enteros y Flotantes:

Sumas: +
Resta: -
Multiplicación: *
División: /
División entera: //
Residuo de una división: %
Potencia y raiz n-esima: **
Mayor que: >
Menor que: <
Igual que: ==
Mayor o igual: >=
Menor o igual: <=
Diferente de: !=

Operaciones con Strings:

Concatenación: +
Indexado: string[índice]

Clase 3

31 de Agosto del 2022

Tipos de Operadores

Los operadores son símbolos reservados especialmente para designar una operación lógica o matemática dentro del entorno de programación. Los tipos de operadores más comunes son:

Operadores de Asignación: Asignan y modifican el valor o la información contenida en una variable, cuyo símbolo es =.

Operadores Aritméticos: Usados para llevar a cabo operaciones aritméticas, algunos de sus símbolos son +, -, *, /, //, %, **.

Operadores Lógicos: Usados para llevar a cabo procesos de selección lógica, cuyos símbolos son “not”, “and” y “or”.

Operadores de Comparación: Usados para comparar propiedades o valores entre variables, cuyos símbolos son >, <, >=, <=, !=, ==.

Operadores de Pertenencia: Denotan la pertenencia o no pertenencia de un valor o variable a un conjunto de las mismas, cuyos símbolos son “in” y “not in”.

Operadores de Conjuntos: Usados para realizar operaciones de conjuntos como agrupación, intersección o exclusión de datos, cuyos símbolos son |, %, -.

Clase 4

2 de Septiembre del 2022

Funciones integradas

Las funciones son un conjunto de instrucciones combinadas para obtener algún resultado para lograr alguna tarea en específica a la que sea llamada la función. Ahora veremos algunas de las funciones integradas en Python.

Funciones de entrada y salida:

input( ): Función de toma de datos de entrada por teclado de la forma:

x=input()

Ejemplo: nombre = input(“ ¿Cual es tu nombre? ”)

print( ): Función de Salida de Información de la forma:

print(x)

Ejemplo: print (“Hola querido usuario”)

format( ): Modifica el formato de un string según un orden y características especificadas.

Funciones de ayuda:

help( ): Muestra información de cualquier función integrada o uso de estructura de datos para el intérprete.

Ejemplos: help (list) - Muestra la ayuda para las listas.

dir( ): Retorna los atributos y métodos válidos de un objeto especificado.

Ejemplo: tupla = (1, 2, 3, 4)
print (dir (tupla))

['__add__', '__class__', '__class_getitem__', '__contains__', '__delattr__', '__dir__', '__doc__', '__eq__', '__format__', '__ge__', '__getattribute__', '__getitem__', '__getnewargs__', '__gt__', '__hash__', '__init__',

'__init_subclass__', '__iter__', '__le__', '__len__', '__lt__', '__mul__', '__ne__', '__new__', '__reduce__', '__reduce_ex__', '__repr__', '__rmul__', '__setattr__', '__sizeof__', '__str__', '__subclasshook__', 'count', 'index']

Aquí nos muestra todos los métodos que existen para trabajar con esta tupla.

type( ): Esta función nos retorna qué tipo de dato es un objeto.

Ejemplos:
type (1, 2, 3)
<class '**int**' >
type ("Agua")
<class '**str**' >

Funciones matemáticas:

abs( ): Retorna el valor absoluto de un número entero.

Ejemplos:
abs(4)
4
abs(-4)
4

round( ): Devuelve un número entero redondeado para un valor numérico.

Ejemplos:
round(1.5)
2
round (3.141592)
3.1416

pow( ): La función calcula la potenciación de base elevado a exponente.

Ejemplo:
pow(2, 3)
8

Funciones de secuencias:

range( ): La función range representa una secuencia inmutable de números enteros.

Ejemplo: for i in range(5, 10):
print(i)
5, 6, 7, 8, 9

enumerate( ): La función enumerate toma una secuencia o un iterador y retorna objeto de tipo enumerado.

Ejemplo: colores = [“rojo”, “verde”, “azul”]
print(colores)
for i, color in enumerate (colores);
print (f “{i}: {color}”)
0: rojo
1: verde
2: azul

zip( ): La función zip devuelve un iterador de tuplas.

Ejemplo: letra = [“a”, “b”, “c”]
numero = [1, 2 ,3 }
info = list(zip(letra, numero))
print (info)
[(‘a’, 1), (‘b’, 2), (‘c’, 3)]

Operaciones con secuencias

len( ): Retorna el número de elementos que contiene un objeto.

Ejemplos: len (“Hola Juan”)
8
len (1, 9, 5)
3

sum ( ): La función sum retorna los elementos sumados de cada secuencia.

Ejemplo: sum (1, 2, 3) = 6

max ( ): La función max retorna el valor más grande del objeto iterable.

Ejemplos: max ([6, 9, 1])
9
max (“python”)
y

min ( ): La función min retorna el valor más pequeño del objeto iterable.

Ejemplos: min ([6, 9, 1])
1
min (“ python”)
h

sorted ( ): La función sorted retorna una lista de elementos de iterables ordenados de menor a mayor.

Ejemplo: sorted ([5, 8 ,6 ,9, 7])
[5, 6, 7, 8, 9]

Clase 5.1

7 de Septiembre del 2022

Condicional if

Las sentencia condicional if nos permite ejecutar si se cumple una condición dada al programa, es decir si una línea de código no es verdadera, esta no se ejecutaría. La sentencia condicional más básica en Python es la sentencia if, tal como la veremos ahora.

Sintaxis:

```
if condición :
#ejecutar código
```

Ejemplos del condicional If

```
x = 15
if x > 10:
        print('x es mayor que 10')
else:
        print('x es menor que 10')
```

x es mayor que 10

```
y = 5
if y == 5:
        print("Es 5")
elif y == 6:
        print("Es 6")
elif y == 7:
        print("Es 7")
```

```
z=8
if z>=10:
        print('z es mayor o igual a 10')

elif z>=5 and z<10:
        print('z se encuentra entre 5 y 10')
else:
        print('z es menor a 5')
```

Clase 5.2

7 de Septiembre del 2022

Ciclo While

El ciclo While se utiliza para llevar a cabo una secuencia de instrucciones programadas de manera repetida mientras que cierta condición sea cumplida, a lo largo de su ejecución, la condición puede ser modificada hasta el punto de se vuelva falsa, momento en el cual el ciclo cerrará su ejecución y dará paso al resto de instrucciones dadas posteriormente al mismo.

Sintaxis:

```
while condición:
    #ejecutar código
    #condición o contador de corte de ciclo
```

Ejemplos del Ciclo While

```
i=20
while i<=50:
    print(i)
    i+=1
```

(Se imprimirán los números del 20 al 50)

```
j=0
while True:
    print('Ciclo ejecutado {} veces' .format(j))

    if j>100:
        print('Contador Superado. Se acabará la ejecución')
        break

ContraseñaUsuario=input('Ingrese su Contraseña: ')
ContraseñaOriginal='abcdefghij'

while ContraseñaUsuario != ContraseñaOriginal:
    ContraseñaUsuario=input('La contraseña es incorrecta.
                             Ingresela nuevamente')
```