Relatoría: Informática II

Juan Esteban Salgado

Objetivos

- Identificar y aprender las herramientas que brinda el lenguaje de programación *python.*
- Aplicar este lenguaje de programación para solucionar problemas de interés aplicados a las matemáticas o física.
- Presentar un proyecto final en el cual se evalúen todas las competencias enseñadas durante el curso.

Marco Teórico

Clase 23-08-2022

- ***Configuración del Entorno de Trabajo***

Para comenzar con un lenguaje de programación se requieren de ciertas adecuaciones al entorno de trabajo para poder aprenderlo mejor. Para esto, como entorno gráfico se maneja la aplicación *"Visual Studio Code"*, en donde se redacta en cada clase ciertas directrices del trabajo a realizar, además de que esta aplicación es capaz de crear aplicaciones tipo *".py"*, las cuales son de vital importancia para empezar a programar en *python.*

Esto como entorno gráfico, pero también surge la necesidad de organizar la información en repositorios o carpetas de acceso online, en las cuales se pueda estar actualizando y posteando los códigos o anotaciones que se tomen en clase. Para realizar esta labor se maneja la página web de *"Github"*, aquí es sencillo enlazarlo con la aplicación *"Visual Studio Code"* brindando así la capacidad de tener nuestro trabajo en la nube. Cabe resaltar que para utilizar este entorno virtual, es necesario crear una cuenta y descargar una herramienta llamada *"Git"* para poder subir nuestro trabajo cómodamente.

- **Visual Studio Code:** "Visual Studio Code es un editor de código fuente desarrollado por Microsoft para Windows, Linux, macOS y Web. Incluye soporte para la depuración, control integrado de Git, resaltado de sintaxis, finalización inteligente de código, fragmentos y refactorización de código." [1]

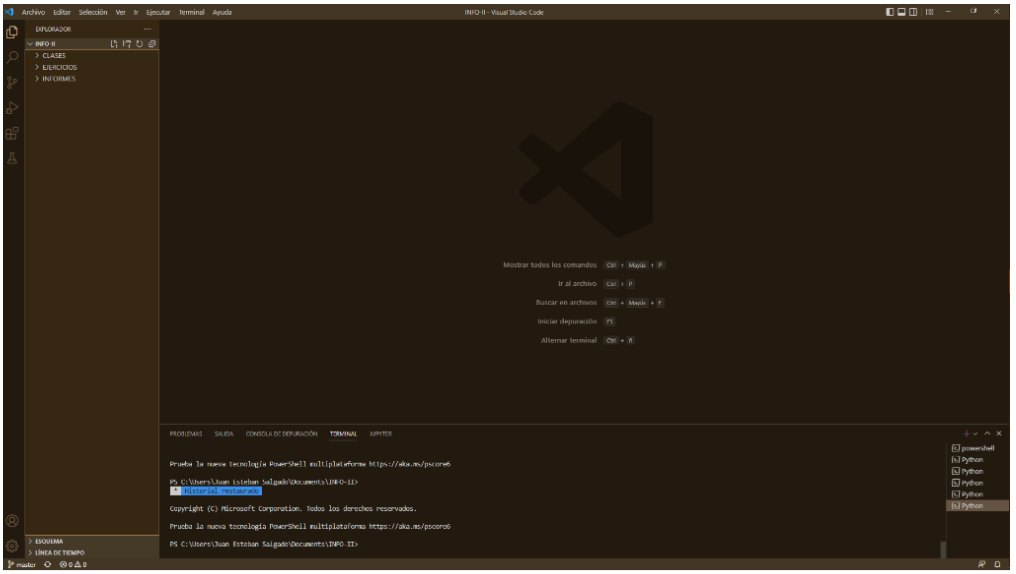

Figura 1. Entorno de desarrollo VSC.

- **GitHub:** “GitHub es una forja (plataforma de desarrollo colaborativo) para alojar proyectos utilizando el sistema de control de versiones Git. Se utiliza principalmente para la creación de código fuente de programas de ordenador.” [2]

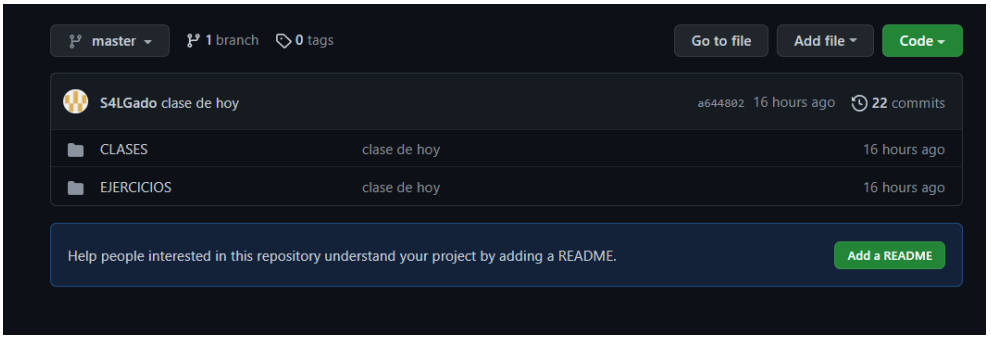

Figura 2. GitHub y los repositorios creados.

Clase 25-08-2022

- ***Creación de Repositorios***

Para subir de forma adecuada nuestros proyectos a la nube, es de vital importancia crear un *repositorio*. Un repositorio se define como una serie de carpetas contenedoras de código o apuntes, las cuales se pueden modificar abiertamente o de forma privada y en las cuales se consignará el trabajo realizado durante el periodo estudiantil. Para crearlo simplemente se debe crear una *carpeta* windows y a través de VSC (Visual Studio Code) se selecciona como repositorio.

Luego de seleccionada se deben hacer unas configuraciones en el terminal de VSC, ya que para que se actualice nuestro trabajo en la nube de GitHub se le debe dar los permisos necesarios al entorno de trabajo. Un *terminal* se define como el lugar en el cual se ejecutan los programas que se van realizando. Se debe escribir lo siguiente:

- **< git config --global user.name "Usuario" >**
- **< git config --global user.email "Correo" >**

Ya con esto es suficiente para comenzar a trabajar en nuestros proyectos, el único detalle que puede ser importante mencionar es a la hora de subir los archivos a GitHub. Para esto nuestros archivos deben pasar por tres fases

- **Changes:** Esta fase consiste en un cambio realizado al código que simplemente ha sido guardado.
- **Staged:** Aquí se confirman los cambios realizados para no subir una versión errónea.
- **Commit:** Simplemente se da la orden de subir los archivos a la nube, para esto es de vital importancia escribir un mensaje corto describiendo el trabajo realizado o en su defecto en qué consiste la actualización enviada.

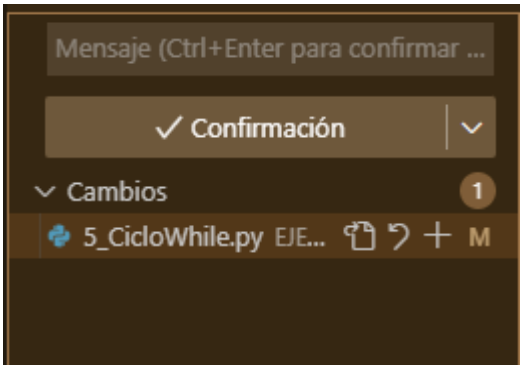

Figura 3. Sección de changes, staged y commit en VSC.

- ***Fundamentos de Python***

Para comenzar con cualquier lenguaje de programación se deben de revisar las características principales del mismo, una que resalta bastante es que *python* es un lenguaje de alto nivel, lo que significa que contiene bastante funciones integradas que pueden ayudar a resolver los problemas de forma más eficaz o rápida. Algunos ejemplos de estas funciones pueden ser: Suma, Resta, Vectores, Condicionales Etc…

Además es un lenguaje *interpretado*, lo que significa que a la hora de ejecutar un programa el lee línea por línea, avisando entonces si hay un error o detenerse en una línea específica.

Posee un *tipado dinámico,* lo cual le da más rigurosidad al lenguaje pues es necesario definir cada tipo de variable o de dato. A continuación se definirán los *tipos de datos:*

- **Tipos de datos**
- **Booleano:** Solo toma valores binarios como verdadero o falso (true, false)
- **Entero:** Números enteros (1, 2, 3).
- **Flotante:** Números racionales (1.2, 3.6).
- **Strings:** Cadenas de caracteres (“hola”, “avión”).
- **Listas:** Serie de elementos cambiables ((1, “hola”)).
- **Tuplas:** Serie de elementos inmutables ([1, “hola”]).
- **Diccionarios:** Permite realizar un mapeo entre dos elementos ((clave:valor), (clave 2:valor)).
- **Conjuntos:** Solo admite valores en una serie de elementos {1,2,3}.

Es claro que para python todo elemento es diferente según como se describe, para esto también existen funciones capaces de cambiar el tipo de dato que se está manejando para que sea más sencillo de tratar.

- **Funciones de conversión de datos**
- **Int:** Convierte un dato en entero.
- **Float:** Convierte un dato en flotante.
- **Str:** Convierte un dato en string (cadena de caracteres).

```
# un entero a un flotante

a = float (a)
print (a, " es del tipo : ", type(a) )

# un flotante a un entero

e = int (e)
print (e, " es del tipo : ", type(e) )

# un string a un entero y flotante

h = int (h)
print (h, " es del tipo : ", type(h) )

h = float (h)
print (h, " es del tipo : ", type(h) )

# un número a un string

a = str (a)
print (a, " es del tipo : ", type(a) )
```

Ejemplo 1. Manejo de las funciones de conversión de datos.

Cabe resaltar que si un dato no cumple con las condiciones no se puede pasar de un tipo a otro, por ejemplo una letra “a” no se puede volver entero o flotante, pues python no sabría describir esta variable numéricamente.

Clase 30-08-2022

- ***Operadores***

Es claro que para resolver un problema se han de necesitar ciertas herramientas matemáticas, algunas básicas y otras un poco más específicas que contiene el lenguaje de python. Los operadores se dividen en diferentes tipos, los cuales se describirán a continuación

- **Tipos de Operadores**

- **Asignación:** = (se usa para declarar una variable).
- **Aritméticos:** + (suma), - (resta), * (multiplicación), / (división), // (división entera), % (residuo de una división), ** (potenciación).

```
# Operadores ariméticos
""" 3 + 9
3.0 + 9
9 ** 0.5
2 ** 32
19 // 2
19 % 3 ***
"hola" + "mundo"
"hola" * 3
["A"] + [1,2,3]
[] + []
(1,2,3) + (1,) """
```

Ejemplo 2. Operadores aritméticos.

- **Lógicos:** and (y), or (o), not (no)

```
# Operadores Lógicos
""" True and True
False and True
False and False
not True
not False
True or True
False or True
1 and 1
0 and 1
1 and 3
1 and "hola"
0 and 3
0 and "hola"
"hola" or "verdadero"
1 or 3"" or "hola"
"" or "false" """
```

Ejemplo 3. Operadores lógicos.

- **Comparación:** > (mayor que), >= (mayor o igual que), < (menor que), <=(menor o igual que), == (igual), != (diferente de).

```
# Operadores de comparación
""" 1 > 2
1 < 3
1 == 1
2 != 1
3 >= 3
5 <= 2
4 > True
True > False
[] > [1,2,3]
"a" > "b" """
```

Ejemplo 4. Operadores de comparación.

- **Pertenencia:** in (adentro), not in (afuera).

```
# Operadores de pertenencia

""" "a" in "abcdefg"
"A" in "ABCDEFG"
1 in [1, 2, 3]
1 in ["1", "2", "3"]
"hola" in "holamundo"
"Hola" not in "holamundo" """
```

Ejemplo 5. Operadores de pertenencia.

- **Conjuntos:** | (unión), & (intersección), - (diferencia).

Tener claro estas definiciones es vital, ya que así se sabe cuál operador es el correcto en cada caso. Cabe resaltar que al utilizar operadores de comparación la respuesta va a ser de tipo booleana, esto es importante ya que a la hora de utilizar condicionales, se puede saber si se cumple o no con la condición dada, para realizar la acción.

Clase 01-09-2022

- ***Funciones Integradas***

Como fue mencionado anteriormente, python se caracteriza por ser un lenguaje de alto nivel, por lo que ahora se describirán las funciones integradas en este lenguaje que facilitan ampliamente el uso del mismo. Además de que brinda un abanico de posibilidades para realizar operaciones de forma rápida y eficaz. A continuación se mencionan algunas de estas funciones y sus aplicaciones:

- **Entrada y Salida:** Sirven para que el usuario pueda ingresar o observar un mensaje en el terminal, Input (entrada de un dato), Print (escribe un mensaje deseado en el terminal), Format(cambia el formato de escritura de un mensaje o variable).

```
"""Edad = int(input("Digite su edad: "))
if Edad >= 18:
| print ("Es mayor de edad")
if Edad < 18:
| print ("Es menor de edad")"""
```

Ejemplo 6. Ejercicio con manejo de input y print.

```
"""numero1 = 0.52941

print(format (numero1, ".0f"))
print(format (numero1, ".2f"))
print(format (numero1, ".5f"))
print(format (numero1, ".1e"))
print(format (numero1, ".2e"))
```

Ejemplo 7. Ejercicio con manejo de format.

- **Ayuda:** Se utilizan para obtener información sobre algún valor o variable, esto con el fin de darle un mejor manejo. Help (da información acerca de un módulo, clase, función o variable en específico), Dir (entrega todas las propiedades del objeto en concreto), Type (indica el tipo de dato que es una variable).
- **Matemáticas:** En este apartado hay bastante funciones, pero las más destacables podrían ser, Abs (valor absoluto), Round (redondea a una cantidad de decimales), Pow (eleva una variable a otra).
- **Conversiones:** Este tipo de funciones ya fueron descritas cuando se mencionaron los tipos de datos, pero a parte de Int, Float y Str. Hay otras más específicas como lo puede ser Complex (convierte un número real en imaginario), Bool (entrega un booleano a partir de una variable cualquiera), List (define un conjunto de datos como

una lista para ser trabajada matemáticamente). Hay una subdivisión muy importante a la hora de las conversiones y esta es entre sistemas numéricos:

- **Bin:** Convierte un número real en binario.
- **Oct:** Convierte un número real en octal.
- **Hex:** Convierte un número real en hexadecimal.

```
# Convertir a binario, octal y hexadecimal
"""
decimal = 9
conversion_binario = bin(decimal)
conversion_octal = oct(decimal)
conversion_hex = hex(decimal)

print(conversion_binario)
print(conversion_octal)
print(conversion_hex ) """

# ¿ Cómo hacer lo contrario ?
"""
bin, oct, hex = "1100110", "146", "66"

print (" bin a decimal: ", int(bin,2))
print (" octal a decimal: ", int (oct,8))
print ("hexadecimal a decimal: ", int(hex,16)) """
```

Ejemplo 8. Manejo de la conversión entre sistemas numéricos.

- **Secuencias:** Estas se utilizan para secuencias numéricas o de variables cualesquiera, Range (genera números enteros hasta que se le de una parada), Enumerate (se utiliza cuando se necesita el valor de un iterable), Zip (entre un objeto comprimido).
- **Operaciones en Secuencias:** a veces es necesario alterar el orden de las secuencias o en su defecto saber algunas propiedades de las mismas. Len (entrega el número de elementos en una secuencia), Sum (entrega la suma de los números en la secuencia), Max (entrega el valor máximo de la secuencia), Min (entrega el valor mínimo de la secuencia).

```
secuencia7 = range(1,10000,3)
lista = [1,2,3,4,5,8,8,9]

print("Tamaño de secuencia", len(secuencia7))
print("Tamaño de lista", len(lista))

print("Mínimo de secuencia", min(secuencia7))
print("Mínimo de lista", min(lista))

print("Máximo de secuencia", max(secuencia7))
print("Máximo de listaa", max(lista))

print("Revertir secuencia", list(reversed(secuencia7)))
print("Revertir lista", list(reversed(lista)))
```

Ejemplo 9. Uso de secuencias y operaciones en secuencias.

Estas son algunas de las operaciones básicas, cabe mencionar que no es necesario llamar a ninguna *librería* para el uso de estas. Más adelante se definirá lo que es una librería y cómo puede traer funciones nuevas que el lenguaje de python no contiene.

Clase 06-09-2022

- ***Condicional If***

Este es el condicional más básico de cualquier lenguaje de programación como C + + o MatLab y en el caso de Python maneja la misma funcionalidad. Este condicional consiste en verificar si una condición dada es verdadera o falsa y luego de eso realizar una serie de tareas por haber cumplido la condición inicial. En caso de que la condición sea falsa simplemente no se ejecuta ninguna acción y se sigue con el resto de líneas del código.

```
if <condicion>:
    <sentencias>
    <sentencias>
    <sentencias>
    <sentencias>

elif <condicion2>:
    <sentencias>
```

Figura 4. Esquema de uso del condicional if.

```
# Ejercicio 1
"""
Pida a un usuario su nombre y su edad. Determine si es mayor de edad,
y muestre un mensaje en pantalla diciendo:
<NOMBRE>, usted es mayor/menor de edad:
"""
"""
nombre = input("INgrese su nombre: ")
edad = int(input("Ingrese la edad: "))

if edad >= 18:
    print(nombre, "{}, usted es mayor de edad", format(nombre))
elif 0 < edad < 18:
    print(nombre, "{}, usted es menor de edad", format(nombre))
"""
# Ejercicio 2
"""
Realice un programa que calcule el mayor de tres numeros
"""
"""
a = float(input("Digite el numero 1: "))
b = float(input("Digite el numero 2: "))
c = float(input("Digite el numero 3: "))

if a >= b and a >= c:
    mayor = a
elif b >= a and b >= c:
    mayor = b
elif c >= a and c >= b:
    mayor = c

print ("El mayor numero entre los ingresados es: ",mayor)

"""
```

Ejemplo 10. Dos ejercicios diferentes en donde se puede aplicar el condicional IF.

Bibliografías o Webgrafías

- [1] Lardinois, Frederic (29 de abril de 2015). «Microsoft Launches Visual Studio Code, A Free Cross-Platform Code Editor For OS X, Linux And Windows». TechCrunch.
- [2] «Microsoft Investor Relations - Acquisitions History». www.microsoft.com (en inglés).
- [3]«Curso de Python». https://codigofacilito.com/cursos/Python