UNIVERSIDAD NACIONAL DE COLOMBIA, SEDE MANIZALES

RELATORÍA 1

DOCENTE CRISTIAN ELÍAS PACHÓN PACHECO

INFORMÁTICA III

ESTUDIANTE
GUSTAVO ADOLFO LÓPEZ ARTEAGA
1002636228

INGENIERÍA FÍSICA
FACULTAD CIENCIAS EXACTAS Y NATURALES

MANIZALES
26/03/2023

Objetivos:

1. Recordar el uso de Python y sus complementos utilizados en el curso de Informática II
2. Afianzar los conocimientos en la creación de funciones lógicas
3. Conocer la plataforma Git y su vinculación con VisualStudioCode

Temario:

1 Creación de Repositorios.

- En esta clase se nos dieron las bases de como eran los pasos para la creación de un repositorio local, desde la creación de la carpeta en el sistema de archivos, hasta iniciarlo en visual en una primera instancia.
- Ya con estos conocimientos previos, nos enseñaron como era el proceso de actualización de este desde el VSC, haciendo énfasis en lo necesario que era no olvidar dejar un mensaje antes del commit
- Para finalizar se nos enseñó como enlazar este mismo con la nube, haciendo la configuración previa del email y el usuario con los siguientes códigos en el terminal

```
* git config --global user.name "juanito"
* git config --global user.email "juanito@unal.edu.co"
```

- Ya después de enlazado, para finalizar la clase, verificamos en Google en Github, que haya quedado creado el repositorio con las respectivas carpetas.

2 Tipos de datos y operadores

- En esta clase empezamos a hacer el repaso de todos los tipos de datos que hay, los cuales íbamos a utilizar en los siguientes trabajos, empezando por los siguientes:

```
Strings:     ""  '' "Hola mundo"  '123456'
Enteros:     -99999999  0  10000000
Flotantes:   100.09    -36.99    0.0
Booleanos:   True False
```

- Dentro del mismo grupo pudimos ver también unos que se iban a utilizar para agrupar los datos de distintas formas:

```
Listas:       []   [0, 1, 2]  ["uno", "dos", "tres"]
Tuplas        ()   (0, 1, 1)  ("uno", "dos", "tres")
Diccionarios  {}   {"hola": "hello", "mundo": "world"}
Conjuntos     {}   {"A", "B", "C", "D"}
```

- De ahí pasamos a ver los distintos operadores y sus respectivas funciones de como los podríamos utilizar, dentro de estos teníamos:

  Por asignación

```
a = 3
b = "hola"
c = [1,2,3]
```

  Aritméticos

```
+, -, *
1/5  => 0.2    division
1//5 => 0      division entera
1%5  => 1      modulo de una division
2**3 => 8      Potencia
2**0.5 =>1.41  Raiz
```

  Lógicos

```
True and True => True
False or False => False
not True  ==> False
```

  Comparativos

```
"hola" == "hola "   =>   False
"hola" != "hola "   =>   True
 3  >  5           =>   False
 3  >= 3           =>   True
-10 <  0           =>   True
-10 <= 10          =>   False
```

  Pertenenecia

```
1 in [1,2,3]      => True
1 not in [1,2,3]  => False
```

3 Built- In Functions

- En esta clase empezamos a familiarizarnos con los diferentes tipos de funciones que podemos utilizar que vienen con el lenguaje como su nombre lo dice, tales como:

  Funciones de entrada y salida:

  Estas funcionan como su nombre lo dice, para ingresar datos en la consola o para mostrarlos en la misma

  ```
  input(), print()
  ```

  Funciones de ayuda:

  Son funciones creadas por el mismo programa para sacarnos de duda con comandos que no recordamos o directamente guiarnos

  ```
  dir(), help()
  ```

  Conversaciones:

  ```
  int(), float(), str(), list(), tuple()
  set(), dict(), complex()..... entre tipos de datos

  hex(), oct(), bin(), int().... entre sistemas numericos
  ```

  Secuencia (listas y tuplas):

  ```
  range(), enumerate(), zip()
  ```

  Operaciones con secuencia

  ```
  len(), sum(), min(), max(), sorted(), map(), filter()

  round()
  ```

4 Métodos

- En esta clase vimos que todo los tipos de datos que se utilizan en Python son objetos, los cuales tienen su respectivo método, lo cual nos facilita la obtención de la información y la actualización de datos, tenemos varios tipos de métodos, como lo son:

  Métodos de los strings:

```
formateo:      capitalize(), upper(), lower()
               title(), center(), strip()

operaciones:   count(subcadena),
               replace(old, new),
               find(subcadena)

verificacion:  isalpha(), isalnum()
               isdigit(), isdecimal()

indexing:      cadena[indice]

slicing:       cadena[inicio:fin:salto]
```

  Métodos de las listas:

```
Operaciones:    append(value), insert(index, value)
                remove(value), pop(index)
                count()

Ordenado:       sort(), reverse()

Almacenamiento: clear(), copy()

Indexado:       [index]

Slicing:        [init:end:step]
```

  Métodos de diccionarios:

```
{
"hola": "hello",
"mundo": "world",
"profesor": "teacher"
}

extracción:     keys(), values(), get(<clave>)
eliminar:       pop(<clave>)
almacenamiento: clear(), copy()
indexado:       diccionario[<clave>]
```

5 Condiciones

- En esta clase tuvimos una introducción a las condiciones, donde se nos explicó, que estas son las instrucciones que debe tener cualquier lenguaje para ejecutar diferentes sentencias dependiendo de una situación, las que vimos son las siguientes:

  Para 1 condición:

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

  Para 2 o más condiciones:

```
if <condicion>:
    <sentencias>
elif <condicion>:
    <sentencias>
else:
    <sentencias>
```

6 Ciclos

- En esta clase, vimos los distintos tipos de ciclos, que nos permitían realizar diferentes procesos dependiendo de una condición, tales como:

Ciclo While:

Este se usa mientras se cumpla una condición, en el momento en el que cambie, este va a parar.

```
while <condicion>:
    <sentencias>
```

Aquí tenemos un ejemplo visto en clase, donde se nos ejemplifica su uso:

```
texto = "habia una vez una í arbitraria"
cont = 0
limite = len(texto)

while cont < limite:
    print(texto[cont], end="--")
    if texto[cont] in "áéíóúÁÉÍÓÚ":
        break
    cont += 1
```

Este va a parar cuando se encuentre una vocal con tilde

Ciclo For:

Este se utiliza, cuando tenemos un ciclo predeterminado para el cual queremos que se realice una acción concreta.

```
for <iterador> in <iterable>:
    <sentencias>
```

Un ejemplo de un uso puede ser:

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

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

7 Funciones

- En esta clase se nos explica el uso de funciones y por que estas son necesarias, ya que nos facilitan mucho al momento de codificar evitar la repetición innecesaria de código, esta se plantea de la siguiente forma:

```
def <nombre_de_la_funcion_>(<parametros>):
    <sentencias>
    return <elemento_a_retornar>
```

Un ejemplo visto en clase es el que utilizamos para saber y analizar si un número es par:

```
def esUnNumeroPar(numero):
    esPar = (numero % 2 == 0)
    return esPar
```

Aquí íbamos a ver los números los cuales, al dividirlos entre 2 tuvieran residuo 0

Conclusión:

Por medio de la práctica y de una metodología lineal, se pudo evidenciar gracias a ejemplos como se pueden utilizar de forma eficaz los diferentes sistemas explicados en este primer corte de repaso de Python, gracias a esto, tuvimos una mejor apropiación de los temas, ya que todos estaban correlacionados y sabíamos con certeza por esto mismo, que les íbamos a dar utilidad en los diferentes ejercicios que íbamos a realizar a partir de vista la teoría.