UNIVERSIDAD TÉCNICA DE MANABÍ

FACULTAD DE CIENCIAS INFORMÁTICAS

TRABAJO: ENSAYO – Informe Github & Python de las listas

Primer Ciclo

PROGRAMACIÓN AVANZADA

VARGAS NOLIVOS HERNÁN PATRICIO

DOCENTE:

GEORGE MICHAEL ALCIVAR BRIONES

ESTUDIANTES:

PERIODO: MAYO – SEPTIEMBRE 2023

Preparación

Para el uso del Git en conjunto con Visual Code es muy fácil se puede seguir la siguiente intrucciones las cuales como resultado nosotros trabajaren en una plantilla con el lenguaje de Python el ejercicio sobre las listas de ese lenguaje.

Requisitos:

Python – Git -Visual Code

Se instalo Git si se configuro la variable global con lo códigos siguientes:

```
git config --list
```

Para ver el estado de git en el momento

```
git config --global user.name "kingyorksh"
git config --global user.email "galcivar4572@utm.edu.ec"
```

Configuramos los datos personales para la configuración de git nombre y correo

```
git config --global -e       # vemos las configuracions
```

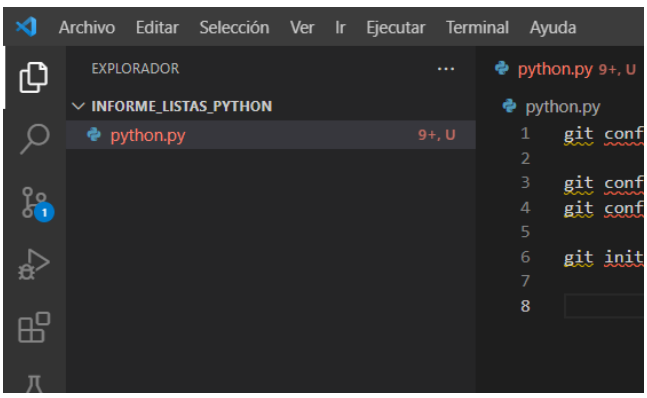
Luego nos situamos en una carpeta desde la app Visual Code e inicializamos o creamos un repositorio local, con el comando:

```
git init
```

Puede utilizarse para convertir un proyecto existente y sin versión en un repositorio de Git, o para inicializar un nuevo repositorio vacío.

Visual Code

Abrimos la carpeta que vamos a usar con el repositorio que acabamos de crear. Creamos un archivo python.py para ver su estado(U de nuevo archivo) y realizamos el código.



```
git add .
git commit -m "Creacion del File python.py de las listas a trabajar"
```

add . significa que los archivos que agregues con este comando estarán listos para ser incluidos en el próximo **commit**.

```
git branch -M main
```

El comando **git branch -M main** se utiliza para renombrar la rama actual del repositorio local a "main".

```
git remote add origin https://github.com/kingyorksh/Informe_Listas_python.git
```

Esto agrega el repositorio remoto con el nombre "origin" y la URL especificada.

```
git push --set-upstream origin main
```

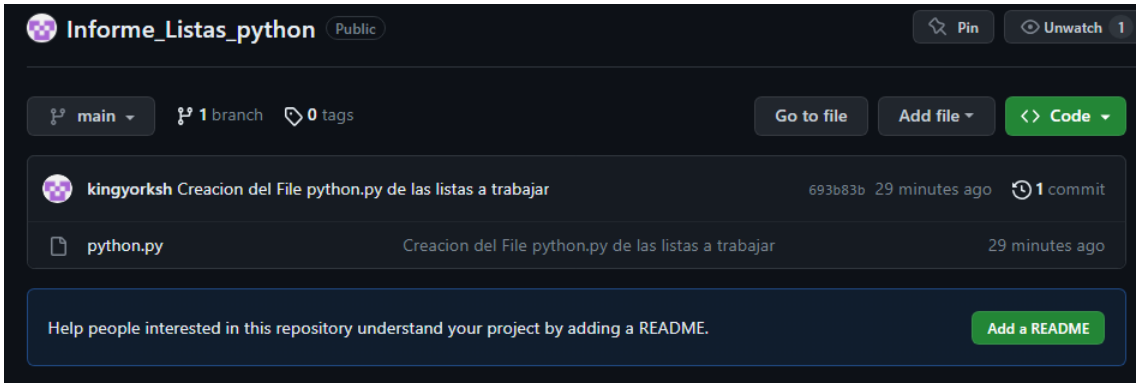
Explicación: se utiliza para enviar tus cambios locales al repositorio remoto y establecer la rama **main** como la rama de seguimiento (upstream branch) en el repositorio remoto llamado **origin**. Esto permite que en futuros **git push** desde la rama **main**, no necesites especificar el nombre de la rama y simplemente puedas ejecutar **git push**.

```
PS C:\Users\gm_ai\OneDrive\Escritorio\Recuperacion\07\PROGRAMCION AVANZADA\Informe_Listas_Python> git push --set-upstream origin main
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 12 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 377 bytes | 377.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/kingyorksh/Informe_Listas_python.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
PS C:\Users\gm_ai\OneDrive\Escritorio\Recuperacion\07\PROGRAMCION AVANZADA\Informe_Listas_Python>
```

Lín. 19, col. 1 Espacios: 4 UTF-8 CRLF Python 3.11

GitHub

Verificamos el repositorio remoto desde la página de github. Ingresando al link donde habíamos creado el repositorio.



Visual Code

Creamos un nuevo archivo donde contendremos el código, que hará de función para ser invocado desde el principal y posteriormente se resubido al github remoto.

Lista.py

```
class Metodos():
    def ingreso(self,lis,tam):
        i=0
        while(i<tam):
            print("Ingrese el [",i,"] valor de la lista")
            num=int(input("numero "))
            lis.append(num)
            i=i+1
    def impresion(self,lis,tam):
        for i in range(tam):
            print(lis[i])
```

Clase llamada "Metodos" con dos métodos: "ingreso" e "impresion". Aquí tienes un resumen de lo que hace cada método:

1. Método "ingreso": Este método recibe dos parámetros, "lis" y "tam", que representan una lista y el tamaño de la lista respectivamente. El método utiliza un bucle while para solicitar al usuario que ingrese valores y los agrega a la lista "lis" utilizando el método append(). El bucle se ejecuta "tam" veces.
2. Método "impresion": Este método recibe dos parámetros, "lis" y "tam", que representan una lista y el tamaño de la lista respectivamente. El método utiliza un bucle for para iterar sobre los elementos de la lista y los imprime en la consola.

Python.py

```
from Listas import Metodos
Listas=[]
_tam=int(input("Ingrese el tamano de la Lista: "))
c=Metodos()
c.ingreso(Listas,_tam)
c.impresion(Listas,_tam)
```

Se importa la clase "Metodos" del archivo "Listas.py". Luego, crea una lista vacía llamada "Listas" y solicita al usuario que ingrese el tamaño de la lista. A continuación, se crea una instancia de la clase "Metodos" llamada "c".

Luego, se llama al método "ingreso" de la instancia "c" y se le pasa la lista "Listas" y el tamaño "_tam" como argumentos. Este método permite al usuario ingresar valores en la lista.

Finalmente, se llama al método "impresion" de la instancia "c" y se le pasa la lista "Listas" y el tamaño "_tam" como argumentos. Este método imprime los valores de la lista en la consola.

Terminal de Visual Code

```
git commit -m "Se creo la clase Metodos() en el archivo Listas"
git push
```