UNIVERSIDAD NACIONAL DE INGENIERÍA

FACULTAD DE CIENCIAS
FUNDAMENTOS DE PROGRAMACION-CC112

ESCUELA DE PROFESIONAL DE CIENCIA DE LA COMPUTACION
PROYECTO FINAL DEL CURSO

SECCIÓN: D **DÍA Y HORA: 28/06/24 1PM-3PM**

Apellidos y Nombres:	**Código:**
Robles Urcia Miguel Angel Sebastian	20221608F
Camarena Frías Ricardo David	20221526J

Nombres de los Docentes:

- Americo Andres Chulluncuy Reynoso

Fecha de entrega del informe: 28/06/24

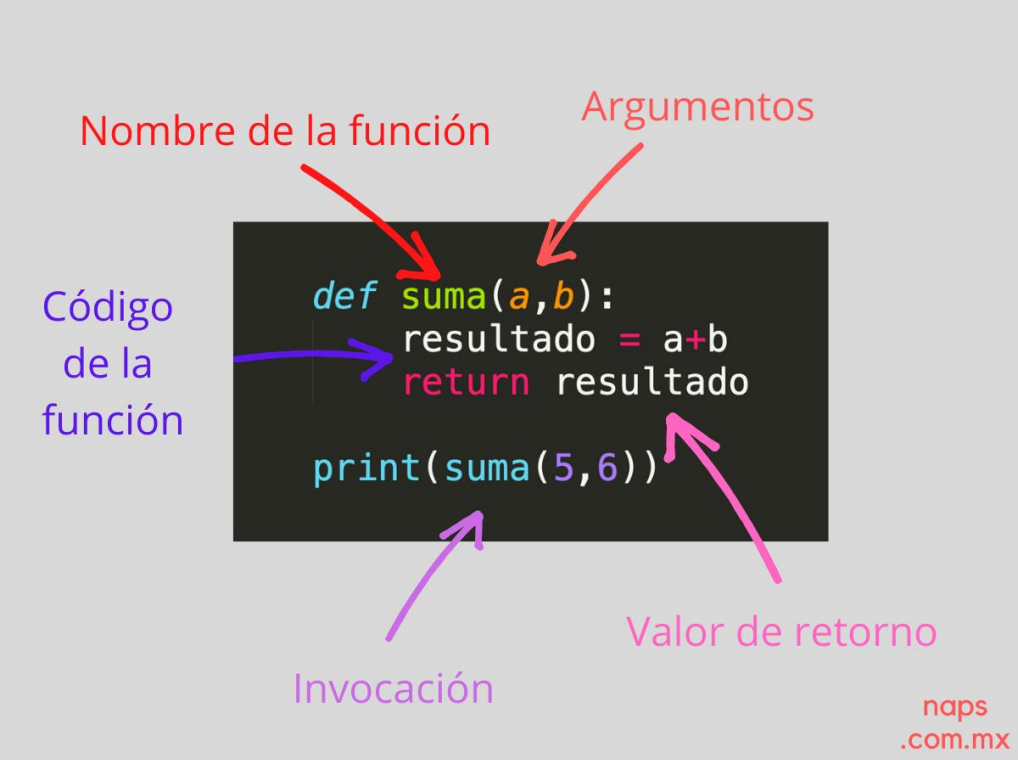
2024-1

1. Introducción

En este informe detallamos el desarrollo de nuestros ejercicios explicando la relación de las ideas hasta la codificación, también exponemos el fundamento teórico utilizado y las conclusiones

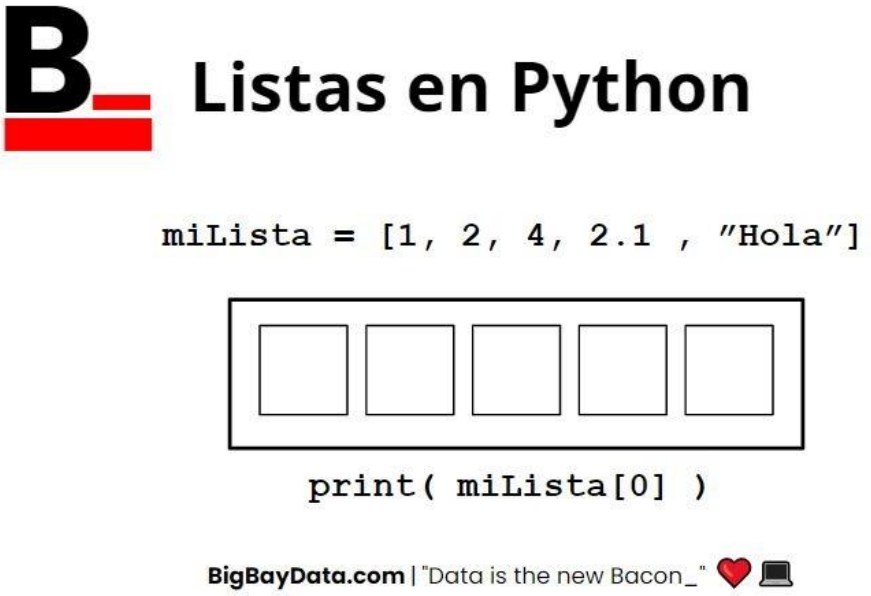
2. Fundamento teórico

Funciones:
Es un código o sentencias, consta del nombre de la función y los parámetros separados por comas.



Cadenas:
Son una secuencia de caracteres se pueden crear utilizando comillas simples o dobles

Listas:
Las listas en python se crean usando corchetes [] y pueden contener datos como enteros, flotantes o cadenas. Las listas en python son dinámicas porque pueden cambiar de tamaño en cualquier momento también son ordenadas y se accede mediante índices.



Diccionarios:
Son estructuras que almacenan pares de clave-valor generalmente son datos que tienen una relación lógica. Se crean usando llaves {}, algunas características es que se pueden añadir mas elementos luego de ser creado

Clases:
Para crear una clase se utiliza class seguidamente el nombre de la clase y dos puntos.

3. Desarrollo

- Parte1

  Ejercicio 1(Funciones):

  Primero definimos una función llamada “Factorial” para cualquier n (entero no negativo) si el numero n ingresado es negativo, el programa te volverá a pedir que digites un numero valido.

  Luego viene el caso Base, donde si el numero ingresado es 0 o 1, el factorial me dará 1.

  El cálculo iterativo si n es mayor que 1, inicializa "resultado” en 1, con el bucle itera desde 2 hasta n, seguido de multiplicar "resultado” por el valor actual de i en cada iteración a continuación, retorna el valor de "resultado” que es el factorial de n

```python
def factorial(n):
    # Verificación de entrada
    if n < 0:
        raise ValueError("El número debe ser un entero no negativo")

    # Casos base
    elif n == 0 or n == 1:
        return 1

    else:
        resultado = 1

        # Cálculo iterativo
        for i in range(2, n + 1):
            resultado *= i

        return resultado
```

  Finalmente tenemos el bucle while para la entrada del usuario.

```python
while True: #Se ejecuta indefinidamente hasta que se ingrese un numero valido

    try: #Intenta convertir la entrada del ususario a un entero

        numero = int(input("Ingrese un número entero no negativo: "))
        if numero < 0:
            print("El número debe ser no negativo. Inténtelo de nuevo.")

        else:
            print(f"El factorial de {numero} es {factorial(numero)}")
            break

    except ValueError: #Si lo de arriba falla, ocurre la excepcion
        print("Por favor, ingrese un número entero válido.")
```

  Ejercicio 2(Cadenas):

Definimos una función llamada “invertir_cadena”, esta función toma un parámetro llamado “cadena” que es el texto que se desea invertir.

Se inicializa la variable “cadena_invertida” como una cadena vacia donde se almacenará la cadena invertida, el bucle for recorre cada carácter en la cadena “cadena”.

En cada iteracion del bucle, el “carácter” actual se agrega al inicio de de la cadena "cadena_invertida”.

Una vez completado el bucle se retorna "cadena_invertida” que contiene la cadena original invertida.

Se utiliza input para pedirle al usuario que ingrese un texto, dicho texto se guarda en la variable "cadena”.

Finalmente, print llama a la funcion “invertir_cadena” con “cadena” como argumento para obtener la cadena invertida.

```python
def invertir_cadena(cadena):

    cadena_invertida = ""  # Se inicializa esta variable como una cadena vacía

    for caracter in cadena:  # Este bucle recorre cada caracter en la cadena de entrada "cadena"
        cadena_invertida = caracter + cadena_invertida  # En cada iteración, "caracter" se agrega al principio de "cadena_invertida"

    return cadena_invertida  # Retorna "cadena_invertida" donde está la cadena original invertida

cadena = input("Ingrese un texto: ") #Se usa el imput para pedirle al usuario que ingrese un texto
print(f"La cadena invertida es: {invertir_cadena(cadena)}")
```

Ejercicio 3:

Definimos una funcion “agregar_elementos”, se inicializa una lista vacia donde se almacenarán los elementos ingresados por el usuario.

Creamos un bucle “while” el cual se ejecutará indefinidamente, hasta que se ejecute "break”, luego con el input solicita al usuario que ingrese un elemento para agregar a la lista.

Con “elemento.lower()” verifica si el usuario ingreso la palabra “salir”, si el usuario decide salir el bucle "while” se interrumpe.

Con “lista,append(elemento)” agrega el elemento ingresado por el usuario a la lista

Luego con “print(f"Elemento '{elemento}' agregado. Lista actual: {lista}")” imprime un mensaje confirmando que el elemento fue ingresado a la lista.

Finalmente, con el "return_lista” la funcion retorna la lista completa con todos los elementos ingresados.

```python
def agregar_elementos():

    lista = []
    while True:
        elemento = input("Ingrese un elemento para agregar a la lista (o 'salir' para terminar): ")
        if elemento.lower() == 'salir':
            break
        lista.append(elemento)
        print(f"Elemento '{elemento}' agregado. Lista actual: {lista}")
    return lista

lista_final = agregar_elementos()
print(f"Lista final: {lista_final}")
```

Ejercicio 4(Diccionarios):

Primero queríamos crear un menú que muestre al usuario 5 opciones para eso simplemente creamos una función que imprima las opciones

```python
# función para mostrar el menú
def menu():
    print("\nMenu:")
    print("1. Buscar un contacto")
    print("2. Agregar un contacto")
    print("3. Eliminar un contacto")
    print("4. Mostrar toda la lista")
    print("5. Terminar el programa")
```

Luego creamos un bucle while para que nos muestre las opciones repetidamente, también usamos if else para indicar que hacer si se selecciona una de las opciones

```
while True:
    menu()
    opcion = input("Seleccione una opción: ")

    if opcion == "1":
        nombre = input("Ingrese el nombre del contacto a buscar: ")
        buscar(nombre)
    elif opcion == "2":
        nombre = input("Ingrese el nombre del contacto: ")
        telefono = input("Ingrese el número de teléfono: ")
        agregar(nombre, telefono)
    elif opcion == "3":
        nombre = input("Ingrese el nombre del contacto a eliminar: ")
        eliminar(nombre)
    elif opcion == "4":
        todo()
    elif opcion == "5":
        print("Programa terminado.")
        break
    else:
        print("Opción no válida. Intente nuevamente.")
```

Luego creamos un diccionario vacío

```
# diccionario de agenda
agenda = {}
```

Luego definimos las funciones que usaremos para cada opción

```
# función para agregar un contacto
def agregar(nombre, telefono):
    agenda[nombre] = telefono
    print(f"Contacto {nombre} agregado.")

# función para buscar un contacto
def buscar(nombre):
    if nombre in agenda:
        print(f"{nombre}: {agenda[nombre]}")
    else:
        print(f"El contacto {nombre} no está en la agenda.")

# funcionpara eliminar un contacto
def eliminar(nombre):
    if nombre in agenda:
        del agenda[nombre]
        print(f"Contacto {nombre} eliminado.")
    else:
        print(f"El contacto {nombre} no está en la agenda.")
```

Ejercicio 5:(Clases)

Primero definimos la clase con los atributos de nombre y edad

```
#creamos una clase
class Persona:
    def __init__(self, nombre, edad):
```

Luego con self almacenamos los atributos

```
self.nombre = nombre
self.edad = edad
```

Luego creamos un método para imprimir los datos

```
#una funcion que sera metodo
    def mostrar(self):
        print(f'Nombre: {self.nombre}, Edad: {self.edad}')
```

- **Parte2**

Primero definimos las funciones que va a hacer nuestra calculadora como suma, resta multiplicación, división, para que estas funciones puedan operar varios números a la vez le pasamos el argumento *args

```
def sumar(*args):
    return sum(args)

def restar(*args):
    resultado = args[0]
    for num in args[1:]:
        resultado -= num
    return resultado

def multiplicar(*args):
    resultado = 1
    for num in args:
        resultado *= num
    return resultado
```

Luego creamos un bucle while, para que las opciones del menú se sigan repitiendo hasta que el usuario decida salir, también imprimimos un menú con todas las operaciones.

```
def calculadora():
    while True:
        print("\nSeleccione una operación:")
        print("1. Suma")
        print("2. Resta")
        print("3. Multiplicación")
        print("4. División")
        print("5. Salir")

        opcion = input("Ingrese su opción (1/2/3/4/5): ")
```

También utilizamos if else, para ejecutar una operación según elija el usuario. En caso de elegir “5” se termina el bucle y finaliza el programa.

```
if opcion == '1':
    print(f"{' + '.join(map(str, nums))} = {sumar(*nums)}")
elif opcion == '2':
    print(f"{' - '.join(map(str, nums))} = {restar(*nums)}")
elif opcion == '3':
    print(f"{' * '.join(map(str, nums))} = {multiplicar(*nums)}")
elif opcion == '4':
    resultado = dividir(*nums)
    if resultado == "Error: División por cero":
        print(resultado)
    else:
        print(f"{' / '.join(map(str, nums))} = {resultado}")
```

- **Parte3(Análisis de datos con Pandas):**

Estos códigos están destinados a crear un mapa utilizando datos geoespaciales y datos de población por regiones en Perú:

Instalación de paquetes:

Aquí se están instalando las bibliotecas necesarias.
Geopandas: Para manejar datos geoespaciales.
Matplotlib: Para trazar gráficos.
Pandas: Para análisis de datos.

```
!pip install geopandas
!pip install matplotlib
!pip install pandas
```

Importación de Bibliotecas:

Se importan todas las bibliotecas necesarias para trabajar con datos geoespaciales(geopandas, cartopy), manipulación de datos (pandas, numpy) y visualización(matplotlib).

```
import matplotlib.pyplot as plt
import numpy as np
import cartopy.crs as ccrs
import pandas as pd
import warnings
import cartopy.feature as cfeature
import geopandas as gpd
import json

%matplotlib inline
warnings.filterwarnings('ignore')
```

Leer datos geoespaciales del Perú:

Lee un archivo GeoJSON (`peru_departamental_simple.geojson`) que contiene los límites geoespaciales de las regiones departamentales del Perú.

```
#Leer mapa vectorial del Perú con regiones
map_df = gpd.read_file('/content/peru_departamental_simple.geojson')
map_df.head()
```

	NOMBDEP	COUNT	FIRST_IDDP	HECTARES	geometry
0	AMAZONAS	84	01	3930646.567	POLYGON ((-77.75893 -6.96451, -77.84586 -6.976...
1	ANCASH	166	02	3596224.600	POLYGON ((-77.31749 -8.53015, -77.28903 -8.589...
2	APURIMAC	80	03	2111415.170	POLYGON ((-72.47177 -14.66140, -72.57725 -14.6...
3	AREQUIPA	109	04	6325588.935	POLYGON ((-75.07333 -15.44294, -75.04965 -15.4...
4	AYACUCHO	111	05	4350381.783	POLYGON ((-74.34595 -12.17374, -74.32187 -12.2...

Leer datos de población por regiones:

Lee un archivo de Excel (`poblacion2020.xlsx`) que contiene datos de población por departamento en Perú.

```
#Leer población por regiones
data_df = pd.DataFrame(pd.read_excel('/content/poblacion2020.xlsx'))
data_df
```

	UBIGEO	DEPARTAMENTO	TOTAL
0	1	AMAZONAS	426806
1	2	ANCASH	1180638
2	3	APURIMAC	430736
3	4	AREQUIPA	1497438
4	5	AYACUCHO	668213
5	6	CAJAMARCA	1453711
6	7	CALLAO	1129854
7	8	CUSCO	1357075
8	9	HUANCAVELICA	365317
9	10	HUANUCO	760267
10	11	ICA	975182
11	12	JUNIN	1361467
12	13	LA LIBERTAD	2016771
13	14	LAMBAYEQUE	1310785
14	15	LIMA	10628470
15	16	LORETO	1027559
16	17	MADRE DE DIOS	173811
17	18	MOQUEGUA	192740
18	19	PASCO	271904
19	20	PIURA	2047954
20	21	PUNO	1237997
21	22	SAN MARTIN	899648
22	23	TACNA	370974

Unir DataFrames:

Une los dos DataFrames (`map_df` y `data_df`) utilizando la columna `NOMBDEP` de `map_df` y `DEPARTAMENTO` de `data_df`.

```
#Unir DataFrames
df_final = map_df.merge(data_df, how = 'inner', left_on = 'NOMBDEP', right_on = 'DEPARTAMENTO', suffixes = ("","_2"))
df_final.head()
```

	NOMBDEP	COUNT	FIRST_IDDP	HECTARES	geometry	UBIGEO	DEPARTAMENTO	TOTAL
0	AMAZONAS	84	01	3930646.567	POLYGON ((-77.75893 -6.96451, -77.84586 -6.976...	1	AMAZONAS	426806
1	ANCASH	166	02	3596224.600	POLYGON ((-77.31749 -8.53015, -77.28903 -8.589...	2	ANCASH	1180638
2	APURIMAC	80	03	2111415.170	POLYGON ((-72.47177 -14.66140, -72.57725 -14.6...	3	APURIMAC	430736
3	AREQUIPA	109	04	6325588.935	POLYGON ((-75.07333 -15.44294, -75.04965 -15.4...	4	AREQUIPA	1497438
4	AYACUCHO	111	05	4350381.783	POLYGON ((-74.34595 -12.17374, -74.32187 -12.2...	5	AYACUCHO	668213

Selección de columnas de interés:

Selecciona las columnas relevantes para el análisis (`DEPARTAMENTO`, `TOTAL` de población y `geometry` para los límites geoespaciales).

```
data_df = data_df[['DEPARTAMENTO', 'TOTAL']]
data_df.head()
```

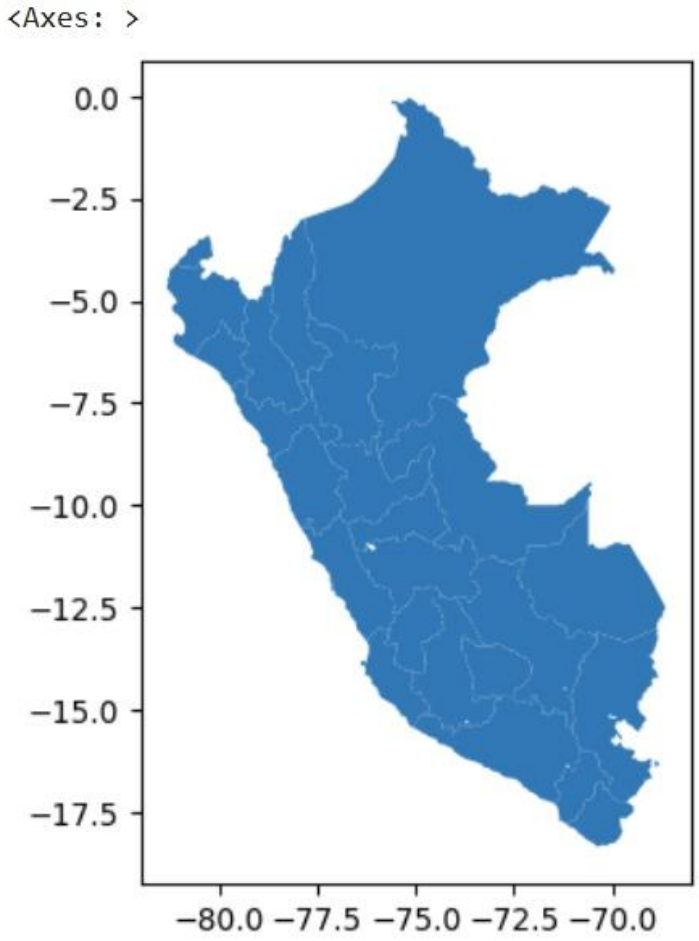
	DEPARTAMENTO	TOTAL
0	AMAZONAS	426806
1	ANCASH	1180638
2	APURIMAC	430736
3	AREQUIPA	1497438
4	AYACUCHO	668213

```
#Las columnas que nos interesan
df_final = df_final[['DEPARTAMENTO', 'TOTAL', 'geometry']]
df_final.plot()
```

Visualización del mapa:

Finalmente, traza el mapa utilizando los datos combinados (`df_final`).

Este proceso combina datos geoespaciales con datos numéricos (en este caso, población) para crear un mapa temático que muestra la distribución de la población en los departamentos de Perú



4. Conclusiones

Python permite interactuar al usuario de forma sencilla, también es fácil integrar los datos, con bibliotecas como mathplotlib y cartopy.