REPUBLICA DOMINICANA

INSTITUTO TECNOLOGICO DE LAS AMERICAS

INTRODUCCION A LA ANALITICA DE DATOS

SECCION:

2025-C-1-2722-3087-TCD-003

MODULO No. VI

INTRODUCCION AL ANALISIS DE DATOS CON PYTHON

Rossy Elania Arvelo Pérez

20240861

Gregory Guillermo De La Rosa

Ejercicios sobre Introducción a Python y Jupyter

1. Instalación de Python

1.1. Explica por qué es importante activar la opción "Add Python to PATH" durante la instalación de Python.

Al instalar Python, es importante activar la opción "Add Python to PATH" por 3 razones importantes:

- Permite ejecutar Python desde cualquier ubicación en la terminal o línea de comandos sin necesidad de especificar la ruta completa del ejecutable.
- Facilita el uso de herramientas y bibliotecas relacionadas con Python sin configuraciones adicionales.
- Evita errores cuando intentas ejecutar Python y el sistema no lo encuentra.

1.2. ¿Cuál es el comando para verificar la versión de Python instalada en tu computadora?

Para verificar la versión de Python instalada se puede utilizar el comando python –version o el mismo comando “versión corta” python -V.

2. Manejo de Jupyter Notebook

2.1. ¿Cuáles son las diferencias entre Jupyter Notebook y Jupyter Lab?

Característica	Jupyter Notebook	Jupyter Lab
Interfaz	Más simple, solo una pestaña por cuaderno	Más avanzada, permite abrir múltiples pestañas y dividir la pantalla
Extensibilidad	Limitada	Permite agregar extensiones y personalizar más opciones
Organización	Un solo cuaderno a la vez	Múltiples cuadernos, archivos, terminales y consolas en una misma vista
Uso recomendado	Para tareas simples o cuando solo necesitas un cuaderno	Para proyectos más complejos y organizados

Jupyter Notebook es más básico y fácil de usar, mientras que **Jupyter Lab** es más avanzado y flexible

2.2. Explica cómo instalar Jupyter Notebook usando pip.

Para instalar Jupyter Notebook usando la terminal o línea de comando se ejecuta el comando “pip install notebook” y para abrirla se utiliza el comando “jupyter notebook”; locual abrirá Jupyter Notebook en su navegador web predeterminado.

3. Ejercicios Prácticos en Jupyter Notebook

- Crea un nuevo notebook en Jupyter y escribe una celda de código que imprima "¡Hola, Python!".
- Usa una celda de Markdown en Jupyter para escribir una breve descripción de lo que es Python.

```
[15]: ## Crea un nuevo notebook en Jupyter y escribe una celda de código que imprima "¡Hola, Python!".
print("Hola Mundo!")
```

```
Hola Mundo!
```

Usa una celda de Markdown en Jupyter para escribir una breve descripción de lo que es Python:

Esta es una descripcion

4. Primeros pasos con Python

- Declara una variable llamada empresa con el valor "Mi Empresa S.A." y otra variable ganancia con el valor 42000000.
- Calcula el 10% de la ganancia como bonos y guárdalo en una variable bono. Luego, imprime el resultado.

```
##Declara una variable llamada empresa con el valor "Mi Empresa S.A." y otra variable ganancia con el valor 42000000.

Empresa = "Mi Empresa S.A."
Ganancia = 42000000
print(Empresa, ganancia)
```

```
Mi Empresa S.A. 42000000
```

```
##Calcula el 10% de la ganancia como bonos y guárdalo en una variable bono. Luego, imprime el resultado.

Bono = ganancia*0.10
print("El monto en bono de este año es de: ",Bono)
```

```
El monto en bono de este año es de:  4200000.0
```

5. Uso de Librerías

- ¿Cómo instalarías la librería pandas en tu entorno de Python?
- Importa pandas en un notebook y usa el comando pd.__version__ para verificar la versión instalada.

Para instalar pandas en debe usar el comando “pip install pandas”. Para importar pandas en el notebook se utiliza el comando “import pandas as pd”.

```
[19]: import pandas as pd
print(pd.__version__)
```

```
2.2.3
```

Ejercicios sobre Series en pandas

1. Creación de una Serie

- Crea una Serie llamada ventas con los siguientes valores: [1500, 2200, 3400, 4100, 2900] y usa los índices ['Enero', 'Febrero', 'Marzo', 'Abril', 'Mayo'].
- Muestra la Serie y verifica su tipo de datos con .dtype.

[23]:
```
ventas = pd.Series([1500, 2200, 3400, 4100, 2900], index=['Enero', 'Febrero', 'Marzo', 'Abril', 'Mayo'])
print(ventas)
```

```
Enero      1500
Febrero    2200
Marzo      3400
Abril      4100
Mayo       2900
dtype: int64
```

2. Operaciones básicas con Series

- Multiplica todos los valores de la Serie ventas por 1.10 para simular un incremento del 10%.
- Filtra y muestra los meses donde las ventas fueron mayores a 3000.

[27]:
```
#Multiplica todos los valores de la Serie ventas por 1.10 para simular un incremento del 10%.
ventas_incrementadas = ventas * 1.10
print(ventas_incrementadas)
```

```
Enero      1650.0
Febrero    2420.0
Marzo      3740.0
Abril      4510.0
Mayo       3190.0
dtype: float64
```

[29]:
```
#Filtra y muestra los meses donde las ventas fueron mayores a 3000.
ventas_mayores_3000 = ventas[ventas > 3000]
print(ventas_mayores_3000)
```

```
Marzo    3400
Abril    4100
dtype: int64
```

3. Acceso a elementos en una Serie

- Muestra las ventas de "Marzo".
- Modifica el valor de "Abril" y cámbialo a 5000.

[34]:
```
#Muestra las ventas de "Marzo"
print("Ventas de Marzo:", ventas['Marzo'])

#Modifica el valor de "Abril" y cámbialo a 5000.
ventas['Abril'] = 5000
print('Abril')
```

```
Ventas de Marzo: 3400
Abril
```

Ejercicios sobre DataFrames en pandas

4. Creación de un DataFrame

- Crea un DataFrame con los siguientes datos:

Producto	Precio	Cantidad
Laptop	1200	10
Monitor	300	25
Teclado	50	50
Ratón	30	75

- Usa pd.DataFrame() para crearlo e imprímelo.

```
#Crea un DataFrame
datos = {
    'Producto': ['Laptop', 'Monitor', 'Teclado', 'Ratón'],
    'Precio': [1200, 300, 50, 30],
    'Cantidad': [10, 25, 50, 75]
}
df = pd.DataFrame(datos)
print(df)
```

```
  Producto  Precio  Cantidad
0   Laptop    1200        10
1  Monitor     300        25
2  Teclado      50        50
3    Ratón      30        75
```

5. Acceso a Datos en el DataFrame

- Muestra solo la columna "Precio".

```
# Mostrar solo la columna "Precio"
print("Columna Precio:")
print(df['Precio'])
```

```
Columna Precio:
0    1200
1     300
2      50
3      30
Name: Precio, dtype: int64
```

- Obtén la fila correspondiente al producto "Teclado".

```
# Obtener la fila correspondiente al producto "Teclado"
print("\nFila correspondiente a 'Teclado':")
print(df[df['Producto'] == 'Teclado'])
```

```
Fila correspondiente a 'Teclado':
  Producto  Precio  Cantidad  Total
2  Teclado      50        50   2500
```

Ejercicios

- Agrega una nueva columna llamada "Total" que sea el resultado de "Precio" * "Cantidad".

```
# Agregar una nueva columna llamada "Total" (Precio * Cantidad)
df['Total'] = df['Precio'] * df['Cantidad']

# Mostrar el DataFrame con la nueva columna "Total"
print("\nDataFrame con la columna 'Total':")
print(df)
```

```
DataFrame con la columna 'Total':
  Producto  Precio  Cantidad  Total
0   Laptop    1200        10  12000
1  Monitor     300        25   7500
2  Teclado      50        50   2500
3    Ratón      30        75   2250
```

6. Filtrado de Datos

- Filtra los productos cuyo precio sea mayor a 100.
- Filtra los productos con más de 20 unidades en stock.

```
# Filtrar productos cuyo precio sea mayor a 100
productos_precio_mayor_100 = df[df['Precio'] > 100]

# Filtrar productos con más de 20 unidades en stock
productos_mas_20_unidades = df[df['Cantidad'] > 20]

# Mostrar los resultados filtrados
print("Productos con precio mayor a 100:")
print(productos_precio_mayor_100)

print("\nProductos con más de 20 unidades en stock:")
print(productos_mas_20_unidades)
```

```
Productos con precio mayor a 100:
  Producto  Precio  Cantidad  Total
0   Laptop    1200        10  12000
1  Monitor     300        25   7500

Productos con más de 20 unidades en stock:
  Producto  Precio  Cantidad  Total
1  Monitor     300        25   7500
2  Teclado      50        50   2500
3    Ratón      30        75   2250
```

7. Ordenación y Estadísticas

- Ordena el DataFrame por la columna "Precio" de mayor a menor.
- Obtén la suma total de todas las unidades disponibles (Cantidad).

```
# Ordenar el DataFrame por la columna "Precio" de mayor a menor
df_ordenado = df.sort_values(by='Precio', ascending=False)

# Obtener la suma total de las unidades disponibles (Cantidad)
suma_unidades = df['Cantidad'].sum()

# Mostrar los resultados
print("DataFrame ordenado por 'Precio' de mayor a menor:")
print(df_ordenado)

print("\nSuma total de las unidades disponibles (Cantidad):", suma_unidades)
```

```
DataFrame ordenado por 'Precio' de mayor a menor:
  Producto  Precio  Cantidad  Total
0   Laptop    1200        10  12000
1  Monitor     300        25   7500
2  Teclado      50        50   2500
3    Ratón      30        75   2250

Suma total de las unidades disponibles (Cantidad): 160
```