Módulo 1: Introducción y manipulación de datos con Python (3 horas)

Presentación del curso y objetivos

Bienvenidos al curso de Python para la Ciencia de Datos. En este curso, aprenderemos a utilizar Python como herramienta principal para la manipulación, análisis y visualización de datos. Los objetivos del curso son:

1. Familiarizarse con Python y su ecosistema de bibliotecas para la ciencia de datos.
2. Aprender a manipular, analizar y visualizar datos utilizando NumPy, Pandas y Matplotlib.
3. Desarrollar habilidades para enfrentar problemas reales de ciencia de datos.

Introducción a Python como lenguaje de programación para la ciencia de datos

Python es un lenguaje de programación versátil, de alto nivel y fácil de aprender. Es ampliamente utilizado en la ciencia de datos debido a su simplicidad y legibilidad, así como a la gran cantidad de bibliotecas disponibles para el análisis y visualización de datos.

```
print("¡Hola, mundo!")
```

Instalación y configuración del entorno de trabajo

Para este curso, necesitarás instalar Python, Jupyter Notebook y las bibliotecas de ciencia de datos más comunes. Puedes seguir los pasos en la guía de instalación para configurar tu entorno.

Introducción a Jupyter Notebook y su uso en la ciencia de datos

Jupyter Notebook es una aplicación web de código abierto que permite crear y compartir documentos que contienen código en vivo, ecuaciones, visualizaciones y texto narrativo. Es muy popular en la ciencia de datos para la exploración, análisis y documentación de datos.

```
# Ejemplo de celda en Jupyter Notebook
import numpy as np
print(np.random.random())
```

Tipos de datos en Python y estructuras de datos básicas

En Python, los tipos de datos básicos incluyen números enteros, flotantes, cadenas de caracteres y booleanos. Las estructuras de datos básicas incluyen listas, tuplas y diccionarios.

```python
# Ejemplo de tipos de datos y estructuras de datos
integer_example = 42
float_example = 3.14
string_example = "Python"
boolean_example = True

list_example = [1, 2, 3, 4, 5]
tuple_example = (1, 2, 3)
dictionary_example = {'one': 1, 'two': 2, 'three': 3}
```

Manipulación de datos con NumPy y Pandas

NumPy es una biblioteca para el lenguaje de programación Python que permite trabajar con matrices y funciones matemáticas de alto nivel. Pandas es otra biblioteca de Python que proporciona estructuras de datos y funciones para trabajar con datos etiquetados y series temporales.

```python
# Ejemplo de manipulación de datos con NumPy
import numpy as np

a = np.array([1, 2, 3, 4, 5])
b = a * 2
print(b)

# Ejemplo de manipulación de datos con Pandas
import pandas as pd

data = {'A': [1, 2, 3], 'B': [4, 5, 6], 'C': [7, 8, 9]}
df = pd.DataFrame(data)
print(df)
```

Funciones y control de flujo en Python

En Python, las funciones son bloques de código reutilizables que realizan una tarea específica. Las funciones se definen usando la palabra clave `def` y se llaman usando su nombre seguido de paréntesis.

```python
def suma(a, b):
    return a + b

resultado = suma(3, 4)
print(resultado)
```

El control de flujo en Python se realiza mediante instrucciones condicionales y bucles. Las instrucciones condicionales incluyen `if`, `elif` y `else`. Los bucles incluyen `for` y `while`.

```python
# Ejemplo de instrucción condicional
x = 42

if x > 50:
```

```
    print("x es mayor que 50")
elif x == 50:
    print("x es igual a 50")
else:
    print("x es menor que 50")

# Ejemplo de bucle for
for i in range(5):
    print(i)

# Ejemplo de bucle while
contador = 0
while contador < 5:
    print(contador)
    contador += 1
```

Lectura y escritura de archivos

Python proporciona funciones integradas para leer y escribir archivos, lo que facilita la importación y exportación de datos.

```
# Lectura de archivo de texto
with open('archivo.txt', 'r') as archivo:
    contenido = archivo.read()
    print(contenido)

# Escritura de archivo de texto
with open('nuevo_archivo.txt', 'w') as archivo:
    archivo.write("Hola, mundo!")
```

Importación y exportación de datos con Pandas

Pandas permite importar y exportar datos en diferentes formatos, como CSV, Excel y JSON.

```
# Importar datos desde un archivo CSV
datos_csv = pd.read_csv('datos.csv')

# Exportar datos a un archivo CSV
datos_csv.to_csv('datos_exportados.csv', index=False)

# Importar datos desde un archivo Excel
datos_excel = pd.read_excel('datos.xlsx', sheet_name='Hoja1')

# Exportar datos a un archivo Excel
datos_excel.to_excel('datos_exportados.xlsx', sheet_name='Hoja1', index=False)
```

Visualización de datos con Matplotlib

Matplotlib es una biblioteca de Python para la creación de gráficos y visualizaciones de datos en 2D y 3D. Permite generar gráficos de líneas, barras, dispersión, histogramas y más.

```python
import matplotlib.pyplot as plt

# Ejemplo de gráfico de línea
x = [1, 2, 3, 4, 5]
y = [2, 4, 6, 8, 10]

plt.plot(x, y)
plt.xlabel('Eje X')
plt.ylabel('Eje Y')
plt.title('Gráfico de línea')
plt.show()

# Ejemplo de gráfico de barras
categorias = ['A', 'B', 'C']
valores = [10, 15, 7]

plt.bar(categorias, valores)
plt.xlabel('Categorías')
plt.ylabel('Valores')
plt.title('Gráfico de barras')
plt.show()
```

Con estos conceptos básicos de Python, estarás listo para explorar aplicaciones más avanzadas en la ciencia de datos y comenzar a trabajar con conjuntos de datos reales.