Introduction to Python for Data Science

Agenda for Week 3

Hands-on Workshop: Python for Data Science

1. Basic Python programming skills with a focus on data analysis.
2. Introduction to Python programming language
3. Basic Python syntax and data structures (lists, tuples, dictionaries)
4. Introduction to Python libraries for data science (Pandas, Numpy)
5. Reading data into Python and performing basic data cleaning
6. Simple data analysis using Python

Part 1: Python Basics

Variables and Data Types

In Python, we can store information in variables. There are several types of data we can store, including integers, floating point numbers, strings, and Booleans.

```
# Integer
x = 10
print(type(x))

# Float
y = 10.0
print(type(y))

# String
z = "Hello, World!"
print(type(z))

# Boolean
a = True
print(type(a))
```

Arithmetic Operations

Python supports all the basic arithmetic operations.

```
# Addition
print(5 + 5)

# Subtraction
print(5 - 2)

# Multiplication
print(3 * 3)
```

```
# Division
print(10 / 2)

# Exponentiation
print(4 ** 2)
```

Logical Operations

Python also supports logical operations, which are often used in conditional statements.

```
# And operation
print(True and False)

# Or operation
print(True or False)

# Not operation
print(not True)
```

Conditional Statements

Conditional statements are used to perform different computations or actions depending on whether a condition evaluates to true or false.

```
# Define a variable
x = 10

# If statement
if x > 0:
    print("x is positive")

# If-else statement
if x % 2 == 0:
    print("x is even")
else:
    print("x is odd")

# If-elif-else statement
if x < 0:
    print("x is negative")
elif x == 0:
    print("x is zero")
else:
    print("x is positive")
```

Loops

Loops are used to repeatedly execute a block of code.

```
# For loop
for i in range(5):
    print(i)

# While loop
i = 0
while i < 5:
    print(i)
    i += 1

# Loop through a list
fruits = ["apple", "banana", "cherry"]
for fruit in fruits:
    print(fruit)
```

Functions

Functions are reusable blocks of code that perform a specific task.

```
# Define a function
def greet(name):
    return "Hello, " + name

# Call the function
print(greet("World"))

# Function with multiple parameters
def power(base, exponent):
    return base ** exponent

# Call the function
print(power(2, 3))
```

Part 2: Python Data Structures

Python has four basic inbuilt data structures: `Lists`, `Tuples`, `Sets`, and `Dictionaries`.

Lists

A list is a collection of items. It is ordered, changeable, and allows duplicate elements.

```
# Define a list
fruits = ["apple", "banana", "cherry"]
print(fruits)

# Access list items by index
```

```
print(fruits[0])

# Change the value of a list item
fruits[1] = "blueberry"
print(fruits)

# Add an item to the list
fruits.append("orange")
print(fruits)
```

Tuples

A tuple is similar to a list, but it is ordered and unchangeable.

```
# Define a tuple
fruits_tuple = ("apple", "banana", "cherry")
print(fruits_tuple)

# Access tuple items by index
print(fruits_tuple[0])

# Trying to change the value of a tuple item throws an error
# fruits_tuple[1] = "blueberry"  # This will throw an error
```

Dictionaries

A dictionary is an unordered collection of key-value pairs.

```
# Define a dictionary
fruit_colors = {
    "apple": "red",
    "banana": "yellow",
    "cherry": "red"
}
print(fruit_colors)

# Access dictionary items by key
print(fruit_colors["apple"])

# Change the value of a dictionary item
fruit_colors["banana"] = "green"
print(fruit_colors)
```

Sets

A set is an unordered collection of unique items.

```

```

```
# Define a set
fruits_set = {"apple", "banana", "cherry", "apple"}
print(fruits_set)  # Duplicates are removed
```

Part 3: Introduction to Pandas and NumPy

Creating arrays in NumPy

```
import numpy as np

# Create a one-dimensional array
a = np.array([1, 2, 3])
print(a)

# Create a two-dimensional array
b = np.array([[1, 2, 3], [4, 5, 6]])
print(b)
```

Manipulating arrays in NumPy

```
# Change an element of the array
b[0, 0] = 10
print(b)

# Get the shape of the array
print(a.shape)
print(b.shape)
```

Creating DataFrames in Pandas

```
import pandas as pd

# Create a DataFrame from a dictionary
df = pd.DataFrame({
    "Name": ["Alice", "Bob", "Charlie"],
    "Age": [25, 30, 35],
    "Occupation": ["Doctor", "Engineer", "Teacher"]
})
print(df)
```

Manipulating DataFrames in Pandas

```

```

```
# Select a column
print(df["Name"])

# Add a new column
df["Salary"] = [100000, 120000, 90000]
print(df)

# Delete a column
df = df.drop("Age", axis=1)
print(df)
```

Reading data from CSV files

```
# Assuming we have a CSV file "data.csv"
df = pd.read_csv("data.csv")
print(df.head())
```

Please note: you'd need to have a `data.csv` file available and replace `data.csv` with the path to your file. Uncomment this section when ready to use.

Basic data exploration

```
# Get the shape of the DataFrame
print(df.shape)

# Get information about the DataFrame
print(df.info())

# Describe the DataFrame
print(df.describe())
```

Conclusion

In this lab, we have covered the basics of Python including data types, arithmetic and logical operations, conditional statements, loops, and functions. We've also explored Python's basic data structures and were introduced to the data science libraries Pandas and NumPy. Practice these concepts and get comfortable with them, as they are the building blocks for more complex data science tasks.

Remember, the aim is to create a learning experience where students can get their hands dirty with code while also understanding the theory behind the actions they perform.

Weekly Meeting Agenda

Each meeting can have its unique structure depending on the nature of the session, but it's useful to have a general framework to ensure meetings are well-organized and efficient. Below is an example of a simple structure for a one-hour meeting:

1. **Introduction** (5-10 minutes):
    - Welcome participants.
    - Share the agenda for the meeting.
    - Recap the last meeting (if applicable) and note any follow-ups.
2. **Main Agenda** (40-45 minutes):
    - This is where the main activities of the meeting will take place.
    - For a **workshop** or **lecture**, this will include the main presentation and demonstration.
    - For a **discussion**, this may involve presenting the topic and then facilitating a group discussion.
    - For a **hackathon** or **project showcase**, this would include the actual work or presentations.
3. **Q&A/Discussion** (5-10 minutes):
    - Reserve some time for attendees to ask questions or discuss the day's topic further.
    - Encourage participation and interaction.
4. **Conclusion and Looking Ahead** (2-3 minutes):
    - Wrap up the meeting and summarize key points.
    - Briefly mention what the next meeting will entail.
    - Thank everyone for their participation.

This is a very flexible structure and can be adjusted to better suit the type and purpose of each meeting.
For instance, a hands-on workshop might require more time for the main agenda, while a planning meeting might have more time allocated to discussion.
The key is to plan and communicate the schedule in advance so that attendees know what to expect and can prepare accordingly.