Introduction to Python 3, In-house Program BSc-Year 1

Dr. Martin Pius

July, 2024

Objectives

- Understand the basic structure of a Python program.
- Learn how to write simple Python scripts.
- Get familiar with Python syntax and basic data types.
- Understand how to control the flow of a Python program.
- Learn how to use conditional statements and loops.
- Learn how to write reusable code using functions.
- Understand the concept of modules and how to use them.
- Learn how to read from and write to files.
- Understand basic data manipulation techniques.
- Final project

Introduction to Python

- **What is Python?**
  - High-level, interpreted programming language.
  - Created by Guido van Rossum and first released in 1991.
- **Why Python?**
  - Easy to learn and use.
  - Widely used in web development, data science, automation, and more.
  - Extensive libraries and community support.
  - https://aloa.co/blog/why-python

Setting Up Python

- **Install Python:**
  - Download from python.org.
- **Install Anaconda (Recommended):**
  - Download from anaconda.com.
- **Choose an IDE:**
  - Jupyter Notebook (comes with Anaconda)
  - VS Code
  - PyCharm

Writing Your First Python Script

- Open your IDE (e.g., Jupyter Notebook).
- Type the following code: `print("Hello, World!")`
- Run the script to see the output.

Basic Syntax

- **Comments:** `# This is a comment`
- **Variables:** `x = 10`

  `name = "Alice"`
- **Data Types:**
  - Integers: `int`
  - Floating point: `float`
  - Strings: `str`
  - Booleans: `bool`

Basic Operators

Arithmetic Operators:

- `x + y # Addition`
- `x - y # Subtraction`
- `x * y # Multiplication`
- `x / y # Division`
- `x % y # Modulus`
- `x ** y # Exponentiation`

Comparison Operators:

- `x == y # Equal to`
- `x != y # Not equal to`
- `x > y # Greater than`
- `x < y # Less than`

Input and Output

Output: `print("Hello, World!")`

Input:

- `name = input("Enter your name:  ")`
- `print("Hello, " + name + "!")`

Activities

- **Activity 1: Print "Hello, World!"**
  - Write and run a script to print "Hello, World!".
- **Activity 2: User Input and Output**
  - Write a script to take user input and display it.
- **Activity 3: Arithmetic Operations**
  - Write a script to perform simple arithmetic operations and print the results.

Summary

- Basic structure of a Python program.
- Writing simple Python scripts.
- Understanding Python syntax and basic data types.
- Using basic operators and handling input/output.

Q&A

Questions?

Additional Resources

- **Books:**
  - "Automate the Boring Stuff with Python" by Al Sweigart
  - "Python Crash Course" by Eric Matthes
- **Online Platforms:**
  - Codecademy
  - Coursera
  - LeetCode
- **Documentation:**
  - Python Official Documentation

Conditional Statements

- **if, elif, else statements:**

```
if condition1:
    statement1
elif condition2:
    statement2
else:
    statement3
```

- Executes `statement1` if `condition1` is true.
- Executes `statement2` if `condition1` is false and `condition2` is true.
- Executes `statement3` if both `condition1` and `condition2` are false.

Loops

- **for loop:**

```
for variable in iterable:
    statement
```

- Repeats `statement` for each item in `iterable`.
- **while loop:**

```
while condition:
    statement
```

- Repeats `statement` while `condition` is true.
-
- **break and continue statements:**

Basic Data Structures

- **Lists:**
  - Creating: `my_list = [1, 2, 3]`
  - Accessing: `my_list[0]`
  - Modifying: `my_list[0] = 10`
- **Tuples:**
  - Creating: `my_tuple = (1, 2, 3)`
  - Accessing: `my_tuple[0]`
- **Dictionaries:**
  - Creating: `my_dict = {'key1': 'value1', 'key2': 'value2'}`
  - Accessing: `my_dict['key1']`
  - Modifying: `my_dict['key1'] = 'new_value'`

Activities

- **Activity 1: Even or Odd**
  - Write a script to determine if a number is even or odd.
- **Activity 2: Print Numbers**
  - Write a script to print numbers from 1 to 10 using a `for` loop.
- **Activity 3: List Manipulation**
  - Create a list and demonstrate accessing and modifying its elements.

Summary

- Control flow with conditional statements.
- Repeating actions with `for` and `while` loops.
- Using `break` and `continue` for loop control.
- Basic data structures: lists, tuples, dictionaries.

Q&A

Questions?

Additional Resources

- **Books:**
  - "Automate the Boring Stuff with Python" by Al Sweigart
  - "Python Crash Course" by Eric Matthes
- **Online Platforms:**
  - Codecademy
  - Coursera
  - LeetCode
- **Documentation:**
  - Python Official Documentation

Functions

- **Defining a function:**

```
def function_name(parameters):
    """Docstring"""
    statement(s)
```

- Example:

```
def greet(name):
    """Greet a person by name."""
    print("Hello, " + name + "!")
```

- **Calling a function:** `greet("Alice")`

Function Parameters and Return Values

- **Parameters:**
  - Positional parameters
  - Keyword parameters
- **Return values:**

```
def add(a, b):
    return a + b
```

- Example:

```
result = add(3, 5)
print(result) # Output:  8
```

Variable Scope

- **Local scope:**
  - Variables defined within a function.
- **Global scope:**
  - Variables defined outside any function.
- Example:

```
global_var = "I am global"
def my_func():
    local_var = "I am local"
    print(global_var)
    print(local_var)
my_func()
print(global_var)
print(local_var) # Error:  local_var is not defined
```

Modules

- **Importing modules:**
  - `import module_name`
  - `from module_name import function_name`
  - `import module_name as alias_name`
- **Using modules:**

```
import math
print(math.sqrt(16)) # Output:  4.0
```

- **Custom modules:**
  - Create a file with `.py` extension.
  - Import it in another script.

Error Handling

- **Using try, except, finally:**

```
try:
     statement(s)
except ExceptionType:
     statement(s)
finally:
     statement(s)
```

- Example:

```
try:
     x = 1 / 0
except ZeroDivisionError:
     print("Cannot divide by zero")
```

Activities

- **Activity 1: Factorial Function**
  - Write a function to calculate the factorial of a number.
- **Activity 2: Using Math Module**
  - Use the `math` module to perform mathematical operations.
- **Activity 3: Custom Module**
  - Write a custom module and import it into another script.

Summary

- Writing reusable code with functions.
- Using parameters and return values in functions.
- Understanding variable scope.
- Importing and using modules.
- Handling errors with `try`, `except`, and `finally`.

Q&A

Questions?

Additional Resources

- **Books:**
  - "Automate the Boring Stuff with Python" by Al Sweigart
  - "Python Crash Course" by Eric Matthes
- **Online Platforms:**
  - Codecademy
  - Coursera
  - LeetCode
- **Documentation:**
  - Python Official Documentation

File I/O

- **Opening and Reading Files:**

```
with open('filename.txt', 'r') as file:
    content = file.read()
```

- **Writing to Files:**

```
with open('filename.txt', 'w') as file:
    file.write('Hello, World!')
```

- **Appending to Files:**

```
with open('filename.txt', 'a') as file:
    file.write('This will be added.')
```

- **Introduction to Pandas:**
  - Powerful library for data manipulation and analysis.
  - Data structures: Series and DataFrame.
- **Reading Data from CSV:**

```
import pandas as pd
df = pd.read_csv('data.csv')
```

Basic Data Manipulation:

- `Selecting: df['column_name']`
- `Filtering: df[df['column_name'] > value]`
- `Sorting: df.sort_values('column_name')`

Data Visualization with Matplotlib

- **Introduction to Matplotlib:**
  - Comprehensive library for creating static, animated, and interactive visualizations.
- **Plotting Basics:**

```
import matplotlib.pyplot as plt
plt.plot([1, 2, 3], [4, 5, 6])
plt.show()
```

- **Creating Different Plots:**
  - Line Plot: `plt.plot()`
  - Scatter Plot: `plt.scatter()`
  - Bar Plot: `plt.bar()`

Data Manipulation Techniques

- **Handling Missing Values:**

```
df.dropna()
df.fillna(value)
```

- **Group By:**

```
grouped = df.groupby('column_name')
grouped.mean()
```

- **Merging DataFrames:**

```
merged_df = pd.merge(df1, df2, on='key')
```

Regular Expressions (Optional)

- **Introduction to Regular Expressions:**
  - Powerful tool for matching patterns in text.
- **Using the `re` Module:**

```
import re
pattern = re.compile('regex_pattern')
matches = pattern.findall('search_string')
```

- **Example:**

```
pattern = re.compile(r'[A-Za-z]+')
text = 'This is a sample text.'
matches = pattern.findall(text)
print(matches) # Output:  ['This', 'is', 'a',
'sample', 'text']
```

Activities

- **Activity 1: File Reading and Writing**
  - Write a script to read content from a file and display it.
  - Write a script to write user input to a file.
- **Activity 2: Data Manipulation with Pandas**
  - Use Pandas to read and manipulate a simple dataset.
- **Activity 3: Data Visualization with Matplotlib**
  - Create different types of plots using Matplotlib.

Summary

- Reading from and writing to files.
- Introduction to Pandas for data manipulation.
- Visualizing data with Matplotlib.
- Basic data manipulation techniques.

Q&A

Questions?

Additional Resources

- **Books:**
  - "Automate the Boring Stuff with Python" by Al Sweigart
  - "Python for Data Analysis" by Wes McKinney
- **Online Platforms:**
  - Codecademy
  - Coursera
  - DataCamp
- **Documentation:**
  - Pandas Documentation
  - Matplotlib Documentation
  - Python re Module Documentation

Objectives of the mini-project

- Explore the California housing dataset.
- Visualize the data using Matplotlib.
- Perform simple linear regression to predict housing prices.

Introduction to the California Housing Dataset

- Understand the dataset's structure and contents.
- Load and inspect the dataset.

```
import pandas as pd
url =
"https://raw.githubusercontent.com/ageron/handson-ml/master
housing = pd.read_csv(url)
print(housing.head())
```

Data Cleaning and Preparation

- Handle missing values.
- Encode categorical variables.
- Split the data into features and target.

```
print(housing.isnull().sum())
housing.fillna(housing.median(), inplace=True)
housing = pd.get_dummies(housing)
X = housing.drop("median_house_value", axis=1)
y = housing["median_house_value"]
```

Data Exploration

- Explore the dataset to understand distributions and relationships.

```
import matplotlib.pyplot as plt
housing.hist(bins=50, figsize=(20, 15))
plt.show()
housing.plot(kind="scatter", x="median_income",
y="median_house_value", alpha=0.1)
plt.show()
```

Data Visualization

- Create informative visualizations to understand data patterns.

```
import seaborn as sns
corr_matrix = housing.corr()
sns.heatmap(corr_matrix, annot=True, cmap="coolwarm")
plt.show()
from pandas.plotting import scatter_matrix
attributes = ["median_house_value", "median_income",
"total_rooms", "housing_median_age"]
scatter_matrix(housing[attributes], figsize=(12, 8))
plt.show()
```

Simple Linear Regression

- Apply linear regression to predict housing prices.
- Evaluate the model.

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
X_train, X_test, y_train, y_test = train_test_split(X,
y, test_size=0.2, random_state=42)
model = LinearRegression()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
rmse = mse ** 0.5
print("Root Mean Squared Error:", rmse)
```

Wrap-Up and Q&A

- Review the day's activities.
- Address any questions or doubts.
- Discuss potential improvements and extensions.

Summary

- Explored the California housing dataset.
- Visualized data using Matplotlib and Seaborn.
- Performed simple linear regression to predict housing prices.

Additional Resources

- **Books:**
  - "Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow" by Aurélien Géron
  - "Python for Data Analysis" by Wes McKinney
- **Online Platforms:**
  - Codecademy
  - Coursera
  - Kaggle
- **Documentation:**
  - Scikit-Learn Documentation
  - Pandas Documentation
  - Matplotlib Documentation