Python Basics

What is Python?

Python is a high-level, interpreted programming language known for its simplicity and readability. It's widely used for web development, data analysis, artificial intelligence, scientific computing, and more. Key features that make Python popular include:

- Readability: Python’s syntax is clear and emphasizes readability, making it easier for developers to write and maintain code.
- Extensive Libraries: Python has a vast standard library and many third-party packages available via PyPI.
- Versatility: It supports multiple programming paradigms, including procedural, object-oriented, and functional programming.
- Community Support: Python has a large and active community, which means plenty of resources, documentation, and support are available.

Examples of Use Cases:
- Web Development: Frameworks like Django and Flask.
- Data Science and Machine Learning: Libraries like Pandas, NumPy, and TensorFlow.
- Automation and Scripting: Writing scripts to automate tasks.
- Scientific Computing: Libraries like SciPy and Matplotlib.
- Game Development: Libraries like Pygame.

Installing Python

Steps to Install Python on Windows:

1. Download the Installer: Go to the [Python official website](https://www.python.org/downloads/) and download the latest installer for Windows.
2. Run the Installer: Double-click the downloaded file and follow the instructions. Ensure you check the box that says “Add Python to PATH”.
3. Verify Installation: Open Command Prompt and type:
sh
```
python --version
```

This should display the installed version of Python.

Steps to Set Up a Virtual Environment:

1. Install virtualenv (if not already installed):
sh
```
pip install virtualenv
```

2. Create a Virtual Environment:
sh
```
python -m venv myenv
```

3. Activate the Virtual Environment:
   - On Windows:
  sh
     myenv\Scripts\activate

   - On macOS and Linux:
  sh
     source myenv/bin/activate

Python Syntax and Semantics

Simple Python Program:

python
print("Hello, World!")

Explanation:
- print() is a built-in function that outputs text to the console.
- The string "Hello, World!" is passed as an argument to the print() function.

Data Types and Variables

Basic Data Types in Python:
- Integer: int
- Floating Point: float
- String: str
- Boolean: bool
- List: list
- Dictionary: dict
- Tuple: tuple
- Set: set

Example Script:

python
# Integer
num = 10
print(num)

# Floating Point
pi = 3.14
print(pi)

# String
greeting = "Hello, Python!"
print(greeting)

```
# Boolean
is_valid = True
print(is_valid)

# List
numbers = [1, 2, 3, 4, 5]
print(numbers)

# Dictionary
person = {"name": "Alice", "age": 25}
print(person)
```

Control Structures

Conditional Statements:

```
python
x = 10
if x > 5:
    print("x is greater than 5")
else:
    print("x is not greater than 5")
```

Loops:

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

# While Loop
count = 0
while count < 5:
    print(count)
    count += 1
```

Functions in Python

Defining and Using Functions:

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

```
# Calling the function
result = add(5, 3)
print(result)
```

Lists and Dictionaries

Differences:
- Lists are ordered collections of items, which can be accessed by their index.
- Dictionaries are unordered collections of key-value pairs, which can be accessed by their key.

Example Script:

```
python
# List
numbers = [1, 2, 3, 4, 5]
numbers.append(6)
print(numbers)

# Dictionary
person = {"name": "Alice", "age": 25}
person["email"] = "alice@example.com"
print(person)
```

Exception Handling

Example of Exception Handling:

```
python
try:
    result = 10 / 0
except ZeroDivisionError:
    print("Cannot divide by zero")
finally:
    print("Execution completed")
```

Modules and Packages

Importing and Using a Module:

```
python
import math

# Using math module
print(math.sqrt(16))
```

File I/O

Reading from a File:

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

Writing to a File:

python
```
lines = ["Hello, World!", "This is a file write example."]
with open('output.txt', 'w') as file:
    for line in lines:
        file.write(line + '\n')
```

These examples cover the basic concepts and operations in Python, from installation to file I/O, providing a solid foundation for further exploration and development.