**Python Basics**

1. **What is Python?** Python is a high-level, interpreted programming language known for its simplicity and readability. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming.

**Key Features:**

* **Easy to Learn and Read:** Python's syntax is clean and straightforward, making it accessible for beginners and experienced programmers alike.
* **Versatility:** It can be used for web development, data analysis, artificial intelligence, scientific computing, and more.
* **Rich Standard Library:** Python comes with a vast standard library that provides modules and packages for various tasks.
* **Interpreted Nature:** Python code is executed line by line by the Python interpreter, which allows for rapid development and debugging.

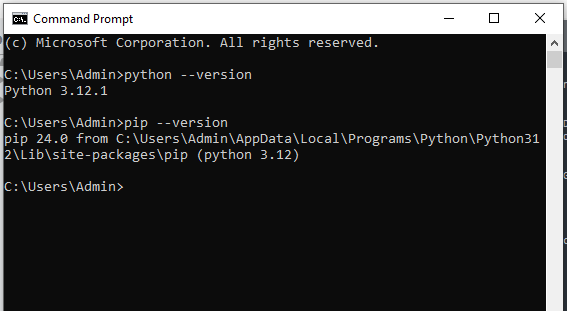
**Examples of Use Cases:**

* Web Development (Django, Flask)
* Data Analysis and Visualization (Pandas, Matplotlib)
* Machine Learning and AI (TensorFlow, PyTorch)
* Scripting and Automation
* Scientific Computing

1. **Installing Python**

**Steps to Install Python:**

1. **Windows:**
   * Download Python installer from [python.org](https://www.python.org/downloads/windows/).
   * Run the installer and check the box "Add Python to PATH".
   * Click "Install Now" and follow the prompts.
   * **Verify Installation:**
   * Open a terminal or command prompt and type python --version or python3 --version to verify.



### Python Syntax and Semantics

**Simple "Hello, World!" Program:**

# hello.py

print("Hello, World!")

 print() is a built-in Python function that outputs the given text to the console.

 "Hello, World!" is a string literal enclosed in double quotes.

1. **Data Types and Variables**

**Basic Data Types in Python:**

1. **Integer (int)**: Whole numbers (e.g., 5, -3).
2. **Float (float)**: Numbers with decimal points (e.g., 3.14, -0.5).
3. **String (str)**: Ordered collection of characters (e.g., "Hello", 'Python').
4. **Boolean (bool)**: Represents truth values True or False.

**Example Script:**

# data\_types.py

# Creating variables of different types

num1 = 5 # integer

num2 = 3.14 # float

name = "Alice" # string

is\_valid = True # boolean

# Printing variables

print(num1)

print(num2)

print(name)

print(is\_valid)

### Control Structures

**Conditional Statements and Loops:**

* **Conditional (if-else) Statement:**

**# conditional.py**

**x = 10**

**if x > 5:**

**print("x is greater than 5")**

**else:**

**print("x is not greater than 5")**

* **For Loop:**

# loops.py

numbers = [1, 2, 3, 4, 5]

for num in numbers:

print(num)

### Functions in Python

**Functions:** Functions in Python are blocks of reusable code that perform a specific task. They improve code readability, modularity, and reusability.

**Example Function:**

**# functions.py**

**# Function to add two numbers**

**def add\_numbers(a, b):**

**return a + b**

**# Calling the function**

**result = add\_numbers(3, 5)**

**print("Sum:", result)**

**# Output: Sum: 8**

### Lists and Dictionaries

**Lists vs Dictionaries:**

* **Lists (list)**: Ordered collection of items accessed by index.
* **Dictionaries (dict)**: Unordered collection of key-value pairs accessed by key.

Example

# lists\_dicts.py

# creating a list of numbers

numbers = [1, 2, 3, 4, 5]

# Creating a dictionary of student grades

grades = {'Alice': 90, 'Bob': 85, 'Charlie': 88}

# Accessing elements

print(numbers[0]) # Output: 1

print(grades['Bob']) # Output: 85

# Adding a new element

numbers.append(6)

grades['David'] = 95

### Exception Handling

**Exception Handling:** Exception handling in Python allows you to handle errors gracefully and prevent program crashes.

**Example:**

# exception\_handling.py

# Handling division by zero

try:

result = 10 / 0

except ZeroDivisionError as e:

print("Error:", e)

finally:

print("Cleanup code") # Runs regardless of exception

### Modules and Packages

**Modules and Packages:**

* **Module**: A file containing Python definitions and statements.
* **Package**: A directory containing Python modules and an \_\_init\_\_.py file.

**Example Using** math **Module:**

# using\_module.py

import math

**#** Using math module to calculate square root

num = 16

sqrt = math.sqrt(num)

print("Square root of", num, "is", sqrt)

### File I/O

**Reading from and Writing to Files:**

* **Reading a File:**

**# read\_file.py**

**# Reading content of a file**

**with open('myfile.txt', 'r') as file:**

**content = file.read()**

**print(content)**

* **Writing to a File:**

# write\_file.py

# Writing a list of strings to a file

data = ['Apple', 'Banana', 'Cherry']

with open('fruits.txt', 'w') as file:

for fruit in data:

file.write(fruit + '\n')

**REFERENCES**

* + 1. Python Crash Course" by Eric Matthes
    2. Automate the Boring Stuff with Python" by Al Sweigart
    3. Stack Overflow Python Tag