Python: Language Basics

Connect with me: Youtube | LinkedIn | WhatsApp Channel | Web | Facebook | Twitter

- Download PDF
- To access the updated handouts, please click on the following link: https://yasirbhutta.github.io/ms-excel/docs/basics.html
- Python Playlist on Youtube
- Download Example Code
- Pyton Resources: Books, Websites, Tutorials
- Python Tools
- Python - Quick Guide for Ultimate Python Beginner's

Want to Learn Python, Join our WhatsApp Channel ✨:

What is Python

- Python is a high-level, general-purpose programming language.
- It is known for its clear syntax, readability, and versatility.
- Python is widely used for `web development`, `data science`, `machine learning`, and `automation`.

Getting Started

- Install Python: Download and install it from https://www.python.org/downloads/.
- Choose a text editor: A program to write code, like Visual Studio Code, Jupyter Notebook, `PyCharm`, or even a simple text editor like `Notepad`.
- Text editor for Android: Pydroid 3 - IDE for Python 3
    - Video: How to: Install Jupyter Notebook on an Android device
- Interactive mode: Experiment with Python directly in your terminal or command prompt using the python command.

> **Important:** Python source code files always use the `.py` extension.

Lesson 1: Python `print` Function

Objectives

- Understand the basic usage of the `print` function.
- Learn how to print different data types.
- Explore advanced `print` function features like formatting and special characters.
- Practice printing in various tasks.

Introduction to `print`

The `print` function is used to output text or variables to the console. or to a file.

Video: Use of print() function in python

Syntax:

```
print(value1, value2, ..., sep=' ', end='\n', file=sys.stdout, flush=False)
```

Parameters:

- `value1, value2, ...`: The values to be printed. Multiple values can be separated by commas.
- `sep`: (Optional) Specifies how to separate multiple values. Default is a space `' '`.
- `end`: (Optional) Specifies what to print at the end. Default is a newline character `'\n'`.
- `file`: (Optional) Specifies the file where to print. Default is `sys.stdout` (console).
- `flush`: (Optional) Specifies whether to forcibly flush the stream. Default is `False`.

Task 1: Basic Printing

Instructions:

1. Print a simple message.
2. Print multiple items separated by commas.

Examples

```
# Task 1.1: Print a simple message
print("Hello, world!")

# Task 1.2: Print multiple items
print("Hello", "world", 2024)
```

Task 2: Printing Different Data Types

Instructions

1. Print integers, floats, and strings.

Examples

```
# Task 2.1: Print different data types
print(42)
print(3.14159)
print("This is a string")
```

Task 3: Using `sep` and `end` Parameters

Instructions

1. Change the separator between printed items.
2. Change the ending character of a print statement.

Examples

```
# Task 3.1: Change the separator
print("apple", "banana", "cherry", sep=", ")

# Task 3.2: Change the ending character
print("Hello", end=" ")
print("world!")

# Task 3.3: Print with a custom ending character:
print("Hello", "World", end="!")
```

Task 4: Print Variables

Instructions

- print variables values using print function

```
# Task 4.1: print a integer variable
x = 5
print(x)

# Task 4.2: print a string variable
message = 'Python is fun'

# print the string message
print(message)

# Task 4.3: print a string variable
message = "How are you?"
print(message)
```

Task 5: String Formatting with f-Strings

Instructions

1. Use f-strings (formatted string literals) for the same purpose.

Examples

```
# Task 5.1: Use f-strings
name = "Ahmad"
age = 30
print(f"My name is {name} and I am {age} years old.")
```

Python f-Strings Explained: What Does print(f'{a=}') Do?

The expression print(f"{a=}") is part of Python's f-string formatting introduced in Python 3.8.

When you use a=, Python prints both the name of the variable and its value. Essentially, it shows what the variable is and its corresponding value.

Task 6: Printing Special Characters

Instructions

1. Print a newline character within a string.
2. Print a tab character within a string.

Examples

```
# Task 6.1: Print a newline character
print("Hello\nWorld")

# Task 6.2: Print a tab character
print("Hello\tWorld")
```

video: Python line break: How to Print Line Break

```
# Task 6.3: Print a tab character

print("Name\tAge\tCity")
print("Alice\t30\tNew York")
print("Bob\t25\tLos Angeles")
```

Output:

```
Name    Age    City
Alice   30     New York
Bob     25     Los Angeles
```

In this example, `\t` is used to align the columns of text.

In Python, the `\t` character is a special escape sequence that represents a horizontal tab. When used in the `print` function or any other string operation, it inserts a tab space in the output. This can be particularly useful for formatting text to make it more readable.

`\t` can be combined with other string manipulation techniques, such as f-strings

```
# Task 6.4: Print a tab character with f-strings

name = "Alice"
age = 30
city = "New York"

print(f"{name}\t{age}\t{city}")
```

Task 7: Printing to a File

Instructions

1. Print a message to a text file instead of the console.

Examples

```
# Task 7.1: Print to a file
with open("output.txt", "w") as file:
    print("Hello, file!", file=file)
```

When you use the instruction with open("output.txt", "w") as file in Python, the file is created in the current working directory of your program. On an Android system, this could be different depending on the environment where the code is executed (e.g., a specific app's data directory, a shared storage location, etc.).

To determine the exact path, you can use the os module to get the current working directory:

```
import os

print(os.getcwd())
```

Regarding file closure, when you use the `with` statement to open a file, Python automatically takes care of closing the file for you once the block of code under the `with` statement is executed. There is no need to

explicitly close the file; it is done automatically when the block is exited. This is one of the benefits of using the with statement for file operations. for more details, see Appendix A

Task 8: Understanding Syntax Errors in Python

Objective: Learn about syntax errors in Python by examining and correcting a sample code snippet.

Instructions:

1. **Review the Code Snippet**: Look at the provided Python code and identify any syntax errors.
2. **Identify the Error**: Understand what a syntax error is and why it occurs.
3. **Correct the Code**: Fix the syntax error in the code snippet.
4. **Explanation**: Write a brief explanation of what the syntax error was and how you corrected it.

Code Snippet:

```
print("Hello World!"
```

Steps:

1. **Review the Code Snippet**: Look carefully at the code snippet above.
2. **Identify the Error**:
    - A syntax error occurs when the Python interpreter finds code that does not conform to the rules of the Python language.
    - The provided code snippet has a syntax error because it has an unmatched parenthesis.
3. **Correct the Code**:
    - To fix the error, ensure all parentheses are properly closed.
    - The corrected code should look like this:

      ```
      print("Hello World!")
      ```

4. **Explanation**:
    - **Syntax Error**: The error was due to a missing closing parenthesis.
    - **Correction**: Adding the closing parenthesis at the end of the print statement fixes the syntax error.

Syntax error:

- A syntax error in programming occurs when the code violates the rules of the programming language's syntax.
- This means that the code's structure and commands do not conform to the expected format that the interpreter or compiler requires to successfully read and execute the code.

See also:

- Invalid Syntax in Python: Common Reasons for SyntaxError

Task 9: Print Your Favorite Quote

- Choose your favorite quote.
- Use Python's `print()` function to display it.
- Ensure the quote is properly formatted (e.g., with quotation marks and correct line breaks if necessary).

Task 10: Create a Simple Receipt**

- Define item names and prices.
- Use `\t` (tab character) to align item names and prices neatly.
- Print a simple receipt showing itemized costs.
- Optionally, include a total cost at the bottom.

Task 11: Use Variables in Print Statements**

- Create three variables: `name`, `age`, and `hobby`.
- Assign appropriate values to each variable.
- Use the `print()` function to create a sentence incorporating these variables.

Task 12: 100 Times "Hello World" Without Loop**

- Print "Hello World" 100 times **without using a loop**.
- Consider string multiplication (`"Hello World\n" * 100`) as a possible solution.

related video: 100 times "hello world" without loop

Task 13: How to Print Multiple Lines**

- Print multiple lines of text using different methods:
  - Using multiple `print()` statements.
  - Using newline characters (`\n`).

Related Video: How to print multiple lines

Task 14: Output to a File**

- Write a short summary of your week (tasks completed, hours worked, achievements).
- Open a text file in write mode (`"w"`).
- Write the summary to the file using Python's `write()` method.
- Close the file properly.

Comments

- Comments are important for making code more readable and understandable, especially for other programmers who may need to read or modify the code.
- Comments in Python are non-executable lines of code and ignored by the Python interpreter when the code is executed.

There are two main types of comments in Python:

- **Single-line comments:** These comments start with the hash symbol (#) and extend to the end of the line.

```
# This is a single-line comment
print("Hello, World!")
```

- **Multi-line comments:** These comments are enclosed in triple quotes (""" or ''').

```
"""
This is a multi-line comment.
It can span multiple lines of code.
"""
print("Hello, World!")
```

See also:

- Video: A Comprehensive Guide to Single Line & Multi-Line Comments

Indentation

In Python, indentation refers to the use of whitespace (spaces or tabs) to denote block-level structure in the code. Python uses indentation to define the scope of code blocks, such as:

- Function definitions
- Loops (for, while)
- Conditional statements (if, elif, else)
- Exception handling (try, except)

In Python, indentation is mandatory and is used to determine the grouping of statements. The number of spaces used for indentation is not fixed, but it's standard to use 4 spaces for each level of indentation. Read more: Indentation - PEP 8 – Style Guide for Python Code

Here's an example:

```
if True:
    print("Hello")  # This line is indented with 4 spaces
    print("World")  # This line is also indented with 4 spaces
```

In this example, the two print statements are indented with 4 spaces, indicating that they are part of the if block.

Python's indentation system has several benefits, including:

- Improved readability: Indentation makes the code structure clear and easy to read.
- Reduced errors: Indentation helps avoid errors caused by mismatched braces or parentheses.

- Simplified syntax: Python's indentation system eliminates the need for explicit block delimiters like braces or keywords.

Another example, consider the following code snippet:

```
if True:
    print("True")
else:
    print("False")
```

Task: 15 Please correct the following Python code:

```
if True:
  print("True")
    print("False")
```

Error message: `IndentationError: unexpected indent`"

See also:

- Indentation in Python - geeksforgeeks.org
- Indentation in Python (With Examples) - askpython.com

Task 16: Code Together, Lead Together

Title: "Teach Your Team"
Instructions:

1. **Each student picks one basic Python topic** (e.g., print function,Basic Printing, Printing Different Data Types, Using sep and end Parameters, Print Variables, String Formatting with f-Strings, Printing Special Characters, Syntax Error, Comments, Indentation).
2. **Prepare a 2-minute explanation** for the group (using examples).
3. **Deliver their explanation** to the team.
4. **Peers ask questions** and give feedback.

Key Learning Points:
✔ Leadership through teaching.
✔ Confidence in public speaking.
✔ Understanding Python better by explaining it.

Python `input()` Function - Lecture Notes

What is `input()`?

The `input()` function in Python is used to take user input from the keyboard. It allows a program to interact with users by asking for information.

Syntax:

```
variable_name = input("Prompt message")
```

- `Prompt message`: A string displayed to the user before they enter input.
- `variable_name`: The variable where the user's input is stored.

Basic Example

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

Explanation:

1. `input("Enter your name: ")` displays the message **"Enter your name: "** and waits for the user to type something.
2. The user types their name and presses Enter.
3. The input is stored in the variable `name`.
4. `print("Hello, " + name + "!")` displays a greeting message with the user's name.

Taking Numerical Input

By default, `input()` returns a string. If you need a number, you must convert it using `int()` or `float()`.

Example:

```
age = int(input("Enter your age: "))
print("In 5 years, you will be", age + 5)
```

Explanation:

1. The user enters their age as a string.
2. `int(input(...))` converts it into an integer.
3. The program calculates the age after 5 years and prints it.

Multiple Inputs

You can take multiple inputs in a single line using `split()`.

Example:

```
name, age = input("Enter your name and age separated by space: ").split()
print("Name:", name)
print("Age:", age)
```

Explanation:

- The user enters **"Alice 25"**.
- `split()` separates it into `name = "Alice"` and `age = "25"`.

Handling Errors

Since `input()` always returns a string, converting it to a number without validation may cause an error.

Example with Error Handling:

```
try:
    age = int(input("Enter your age: "))
    print("You entered:", age)
except ValueError:
    print("Invalid input! Please enter a number.")
```

This prevents crashes if the user enters non-numeric data.

Key Takeaways

- `input()` is used to take user input.
- It always returns a string; convert it when necessary (`int()`, `float()`).
- Use `split()` to take multiple inputs.
- Handle errors using `try-except` to avoid crashes.

Would you like me to include exercises for practice? 😊

Fix the Error

Question 1: Fix the Parenthesis Error

Task: Identify and fix the syntax error in the following code snippet.

```
print("Hello World!"
```

Error: SyntaxError: The closing parenthesis is missing.

Corrected Code:

```
print("Hello World!")
```

Question 2: Fix the Indentation Error [Python Quiz #67]

Task: Fix the indentation error in the following code snippet.

```
if True:
  print("Hello")
    print("World")
```

Error: `IndentationError: unexpected indent` at the second print statement.

Question 3: Fix the Incorrect Variable Usage Error [Python Quiz 68]

Task: Fix the error where the variable name is incorrectly used.

```
name = "Ahmad"
print(f"My name is {namee} and I am 30 years old.")
```

Error: `NameError: name 'namee' is not defined` because `namee` is not the correct variable name.

Question 3: Fix the Missing Import Error

Task: Correct the following code by adding the necessary module import.

```
print(os.getcwd())
```

Error: `NameError: name 'os' is not defined` because the `os` module was not imported.

Corrected Code:

```
import os
print(os.getcwd())
```

Question 5: Fix the File Output Error

Task: Fix the syntax issue in printing to a file.

```
with open("output.txt", "w") as file:
print("Hello, file!", file=file)
```

Error: `IndentationError: expected an indented block` because the print statement is not indented.

Corrected Code:

```
with open("output.txt", "w") as file:
    print("Hello, file!", file=file)
```

True/False (Mark T for True and F for False)

1. An indentation error in Python is considered a syntax error.

Answer Key (True/False):

1. True

Multiple Choice (Select the best answer)

Python

1. **What is Python?**
    - A) A type of snake
    - B) A high-level programming language
    - C) A web browser
    - D) An operating system

    Answer: B) A high-level programming language

2. **Which of the following is a popular Integrated Development Environment (IDE) for Python?**
    - A) Visual Studio Code
    - B) Microsoft Word
    - C) Adobe Photoshop
    - D) Google Chrome

    Answer: A) Visual Studio Code

3. **Which tool is used for managing and installing Python packages?** [Python Quiz #53]
    - A) pip
    - B) npm
    - C) yarn
    - D) gem

    Answer: A) pip

4. **What does IDE stand for in the context of Python programming?**

   - A) Integrated Development Environment
   - B) Integrated Data Environment
   - C) Internal Development Engine
   - D) Internet Development Environment

   Answer: A) Integrated Development Environment

5. **What is the purpose of virtual environments in Python?**

   - A) To provide a web-based interface for Python development
   - B) To create isolated environments for different Python projects
   - C) To compile Python code into machine code
   - D) To improve the speed of Python code execution

   Answer: B) To create isolated environments for different Python projects

6. **Which tool is used for version control in Python projects?**

   - A) Git
   - B) Docker
   - C) Kubernetes
   - D) Jenkins

   Answer: A) Git

7. **Which of the following is the correct extension of the Python file?** [Python Quiz #51]

   - A) .python
   - B) .pl
   - C) .p
   - D) .py

8. **What is Jupyter Notebook primarily used for in the Python ecosystem?** [Python Quiz #54]

   - A) Web development
   - B) Writing and sharing code, visualizations, and text
   - C) Game development
   - D) Mobile app development

   Answer: B) Writing and sharing code, visualizations, and text

9. **Who created Python and in which year?** [1]

   - A) James Gosling in 1995
   - B) Guido van Rossum in 1991
   - C) Bjarne Stroustrup in 1983
   - D) Dennis Ritchie in 1972

   Answer: B) Guido van Rossum in 1991

10. **Which of the following features is a key characteristic of Python?** [Python Quiz #52]
    - A) Statically typed
    - B) Compiled language
    - C) Interpreted language
    - D) Requires manual memory management

11. **What is the purpose of the virtual environment tool in Python?**
    - A) To create virtual machines
    - B) To manage and isolate project-specific dependencies
    - C) To design virtual reality experiences
    - D) To develop mobile applications

Answer: B) To manage and isolate project-specific dependencies

12. **What is the Python Package Index (PyPI)?**
    - A) A text editor for Python
    - B) A library for machine learning
    - C) A repository of software for the Python programming language
    - D) An IDE for Python development

Answer: C) A repository of software for the Python programming language

13. **What is `pip` in Python?**
    - A) A text editor
    - B) A package installer
    - C) A version control system
    - D) A web framework

Answer: B) A package installer

14. **Which tool is commonly used for creating virtual environments in Python?**
    - A) npm
    - B) pip
    - C) venv
    - D) docker

Answer: C) venv

15. **Which Integrated Development Environment (IDE) is specifically designed for Python development?** [Python Quiz #54]
    - A) Visual Studio
    - B) IntelliJ IDEA
    - C) PyCharm
    - D) Eclipse

Answer: C) PyCharm

16. **Which of the following is true about Python?**
    - A) Python is a compiled language

- B) Python is a low-level programming language
- C) Python supports object-oriented programming
- D) Python does not support modules

Answer: C) Python supports object-oriented programming

17. **Which of the following is NOT a feature of Python?**
    - A) Interpreted language
    - B) Compiled language
    - C) High-level language
    - D) Object-oriented language

Answer: B) Compiled language

print function

video:Can You Guess the Output of this Python Code? | print Quiz

18. **What is the output of the following code?** [Python Quiz #28]

Code:

```
print("I", "love", "Python", sep="-")
```

Options:

- A) I-love-Python
- B) I love Python
- C) I, love, Python
- D) Syntax error

Watch the video for the answer: https://youtube.com/shorts/WD92M8WXRZM?si=kJ5jbIAaIlJ_63ia

19. **What will be the output of the following code?** [Python Quiz #29]

```
x = 5
y = 3
print("The value of x is", x, "and the value of y is", y)
```

- A) Syntax error
- B) The value of x is {} and the value of y is {}
- C) The value of x is 3 and the value of y is 5
- D) The value of x is 5 and the value of y is 3

Watch the video for the answer: https://youtube.com/shorts/ZE2yfAJsCxk?si=6UvXfKLmR56c-Qu9

20. **Which of the following is the correct syntax for the print statement in Python?**

- A) print ("text")
- B) println ("text")
- C) echo ("text")
- D) None

21. **What will be the output of the following code?**

```
print("Hello, world!")
```

- A) Hello
- B) world
- C) Hello, world!
- D) There will be no output.

22. How can you print multiple values on a single line in Python?

23. ☐ Use commas to separate the values within the print statement.

24. ☐ Use semicolons to separate the values within the print statement.

25. ☐ Use the + operator to concatenate the values before printing.

26. ☐ Create a list of the values and print the list.

27. Which of the following statements will print the value of the variable x?

28. ☐ print(x)

29. ☐ print "x"

30. ☐ println(x)

31. ☐ echo x

32. What is the purpose of the sep argument in the print function?

33. ☐ To specify the separator between multiple values printed on the same line.

34. ☐ To specify the end character for the printed line.

35. ☐ To specify the file to which the output should be printed.

36. ☐ To specify the format of the output.

37. What is the purpose of the end argument in the print function?

38. ☐ To specify the separator between multiple values printed on the same line.

39. ☐ To specify the end character for the printed line.

40. ☐ To specify the file to which the output should be printed.

41. ☐ To specify the format of the output.

42. How can you print a string without a newline character?

43. ☐ print(string, end="")

44. ☐ print(string, sep="")

45. ☐ print(string + "")

46. ☐ print(string; "")

47. **What is the output of the following code?** [Python Quiz #75]

```
name = "Alice"
age = 30
print("My name is", name, "and I am", age, "years old.")
```

- A) My name is Alice and I am 30 years old.
- B) My name is Aliceand I am 30years old. (No separation)
- C) Alice 30 (Values printed without labels)
- D) An error (Incorrect syntax)

28. How can you format a string in Python to insert variables directly within it?

- a) Using string concatenation with the + operator (Limited control)
- b) Using the format method (Less readable for complex formatting)
- c) Using f-strings
- d) All of the above (f-strings are generally preferred)

29. Which of the following statements is NOT valid for the print function in Python?

- a) print("Hello, world!")
- b) print(5 + 3) (prints the result of the expression)
- c) print() (prints an empty line)
- d) print(x, y, sep=",") (prints x and y with a comma separator)

30. How can you prevent a newline character from being printed at the end of the output in Python?

- a) By using a `semicolon` at the end of the print statement (this is not valid Python syntax)
- b) Using the `end` argument within the print function and setting it to an empty string ("")
- c) Specifying a special flag in the function call
- d) There's no way to suppress the newline character

31. **What is a syntax error in Python?**

- A) An error caused by incorrect logic in the code.
- B) An error detected when the code violates the rules of the Python language.
- C) An error that occurs during runtime.
- D) An error caused by a variable not being defined.

32. **Identify the error in the following code snippet:**

```
prin("Hello World!")
```

- A) Incorrect variable name
- B) Missing colon
- C) Misspelled keyword
- D) Missing parenthesis

Hint NameError: name 'prin' is not defined.

33. **What should be done to correct the syntax error in the following code?**

```
print("Hello World!"
```

- A) Add a closing quotation mark.
- B) Add a closing parenthesis.
- C) Add a colon at the end.
- D) Indent the line correctly.

Comments:

34. What is the primary purpose of comments in Python code?
35. ☐ To execute instructions for the computer
36. ☐ To temporarily disable lines of code
37. ☐ To make the code more readable and understandable for humans
38. ☐ To create errors for debugging
39. Which of the following is the correct syntax for a single-line comment in Python?
40. ☐ // This is a comment
41. ☐ /* This is a comment */
42. ☐ # This is a comment
43. ☐ { This is a comment }
44. How can you create a multi-line comment in Python?
45. ☐ Using triple single quotes (''')
46. ☐ Using triple double quotes (""")
47. ☐ Using backslash () at the end of each line
48. ☐ Using the comment keyword

49. What happens when you run code that includes comments?

50. ☐ The comments are executed along with the code.

51. ☐ The comments are ignored by the Python interpreter.

52. ☐ The comments are displayed as output.

53. ☐ The comments are converted into machine code.

Indentation:

38. What is the purpose of indentation in Python?
39. ☐ To define code blocks
40. ☐ To define functions
41. ☐ To define variables
42. ☐ To print output

Answer: a) To define code blocks

39. Which of the following is a correct way to indent code in Python?

- a) Using tabs
- b) Using spaces
- c) Using both tabs and spaces
- d) Using neither tabs nor spaces

Answer: b) Using spaces (Python recommends using 4 spaces for indentation)

40. What happens if you don't indent your code in Python?

- a) It will run correctly
- b) It will throw a syntax error
- c) It will print an error message
- d) It will ignore the code

Answer: b) In Python, improper indentation specifically results in an IndentationError. While a syntax error is a broad category for any error in the syntax, an IndentationError is a specific type of syntax error related to incorrect indentation.

41. What is the indentation error in the following code?

```
if True:
  print("True")
    print("False")
```

- a) Missing indentation
- b) Extra indentation
- c) Incorrect indentation
- d) No error

Answer: b) Extra indentation (the second print statement has extra indentation)

43. What is the standard number of spaces used for indentation in Python? a) 2 spaces b) 4 spaces c) 6 spaces d) 8 spaces

Answer: b) 4 spaces

44. Which of the following code snippets is correctly indented?

a)
```python
if True:
  print("True")
    print("False")
```

b)

```python
if True:
    print("True")
    print("False")
```

c)

```python
if True:
     print("True")
print("False")
```

d)

```python
   if True:
  print("True")
      print("False")
```

Answer: b)

```python
 if True:
    print("True")
    print("False")
```

45. In Python, indentation is used to define _____ in the code.

- A) Loops
- B) Functions

- C) Classes
- D) Blocks of code

Exercises

Print Functions

1. **Basic Printing**
   - Write a Python program to print the following:

     ```
     Hello, World!
     ```

   Solution:

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

2. **Printing Variables**
   - Create two variables, `name` and `age`, and print them using the `print` function. Use the variables to print the following:

     ```
     My name is John and I am 25 years old.
     ```

   Solution:

   ```
   name = "John"
   age = 25
   print("My name is", name, "and I am", age, "years old.")
   ```

3. **Formatted Strings**
   - Use an f-string to print the following using the variables `name` and `age`:

     ```
     My name is John and I am 25 years old.
     ```

   Solution:

   ```
   name = "John"
   age = 25
   print(f"My name is {name} and I am {age} years old.")
   ```

Indentation

4. **Correct Indentation**
   - The following code has incorrect indentation. Fix it so that it runs correctly:

```
def greet(name):
print(f"Hello, {name}!")

greet("Alice")
```

Solution:

```
def greet(name):
    print(f"Hello, {name}!")

greet("Alice")
```

5. **Conditional Indentation**
   - Write a Python program that checks if a number is positive, negative, or zero and prints the result. Ensure proper indentation is used.

Solution:

```
number = int(input("Enter a number: "))

if number > 0:
    print("The number is positive.")
elif number < 0:
    print("The number is negative.")
else:
    print("The number is zero.")
```

Comments

6. **Single-Line Comments**
   - Add comments to the following code explaining each line:

```
name = "Alice"
age = 30
print(f"{name} is {age} years old.")
```

Solution:

```
# Assign the string "Alice" to the variable 'name'
name = "Alice"

# Assign the integer 30 to the variable 'age'
age = 30

# Print the name and age using an f-string
print(f"{name} is {age} years old.")
```

7. **Multi-Line Comments**
    - Use a multi-line comment to describe what the following code does:

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

result = add(3, 5)
print(result)
```

Solution:

```
"""
This program defines a function 'add' that takes two arguments 'a' and 'b'
and returns their sum. It then calls this function with the arguments 3
and 5, stores the result in the variable 'result', and prints the result.
"""

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

result = add(3, 5)
print(result)
```

Review Questions

1. Explain what comments and indentation are in Python, and give examples to show how they work.

References and Bibliography

[1] “Guido van Rossum,” Wikipedia, May 24, 2021. https://en.wikipedia.org/wiki/Guido_van_Rossum

- Indentation in Python - geeksforgeeks.org
- Indentation in Python (With Examples) - askpython.com

Appendices

Appendix A: os.getcwd()

`os.getcwd()` is a function in Python that returns the current working directory (CWD) of the script. The "current working directory" is the folder from which your Python script is running. It is useful for figuring out the exact location of the script during execution.

Explanation:

- `os`: This is the module in Python that provides a way to interact with the operating system.
- `getcwd()`: This function within the `os` module returns a string representing the current working directory.

Example:

```
import os

# Get the current working directory
current_directory = os.getcwd()

# Print the current working directory
print("Current Working Directory:", current_directory)
```

Output: If you run this script, you might get an output like this (depending on where the script is being executed):

```
Current Working Directory: /Users/yourusername/Documents/my_project
```

This tells you that the script is being run from the `/Users/yourusername/Documents/my_project` directory.

Use Case: Imagine you are writing a script that needs to read or write files. Knowing the current working directory will help ensure you reference the correct paths to those files.

**Appendix B: Compile time and Run time

Compile time and run time refer to different phases in the lifecycle of a program:

Compile Time:

Definition: This is the phase when the source code is converted into machine code (or bytecode) by a compiler.

Errors: Errors that occur at compile time are typically related to syntax or static type checks (in statically typed languages). These include things like syntax errors, missing semicolons, or mismatched data types.

Duration: This happens before the program is executed. In languages like C, Java, and C++, the code is compiled before it is run.

Key Point: The program cannot run if there are compile-time errors.

Run Time:

Definition: This refers to the phase when the compiled code is executed by the machine or interpreter.

Errors: Run-time errors occur while the program is being executed. These include logical errors, exceptions, or any unexpected input or environment conditions (like division by zero or file not found errors).

Duration: This happens after the compilation phase (in compiled languages) or during interpretation (in interpreted languages like Python).

Key Point: The program is actively running and performing tasks.

Summary:

Compile time is when the code is translated into machine code before execution.

Run time is when the code is actively being executed on the machine.