**Lesson 17: Understanding Basic Python Code**

In this lesson, we will break down a short piece of code line by line, explaining each part and its purpose. This code demonstrates how to work with strings in Python.

first\_name = "ahmed"

last\_name = 'sami'

# print(type(last\_name))

print(first\_name + " " + last\_name)

**Line 1: Variable Assignment**

first\_name = "ahmed"

* **Explanation**: This line declares a variable named first\_name and assigns it the string value "ahmed".
* **Purpose**: In programming, variables are used to store data that can be referenced later. Here, we are storing the first name of a person.
* **Example**: If we change the value to first\_name = "John", now first\_name will hold "John" instead of "ahmed".

**Line 2: Variable Assignment**

last\_name = 'sami'

* **Explanation**: Similar to the first line, this line creates another variable named last\_name and assigns it the string value 'sami'.
* **Purpose**: This variable holds the last name of a person. Using single quotes or double quotes in Python for strings is interchangeable.
* **Example**: If we had last\_name = "Doe", then last\_name would now contain "Doe".

**Line 3: Comment**

# print(type(last\_name))

* **Explanation**: This line is a comment, which is ignored by the Python interpreter. It is used to provide explanations or notes within the code.
* **Purpose**: Comments help other developers (or yourself in the future) understand what the code is doing. The commented line, if uncommented, would print the type of the variable last\_name.
* **Example**: If we remove the # and run print(type(last\_name)), it would output <class 'str'>, indicating that last\_name is a string.

**Line 4: Print Statement**

print(first\_name + " " + last\_name)

* **Explanation**: This line outputs the combined value of first\_name and last\_name to the console, with a space in between.
* **Purpose**: The print() function is used to display information to the user. Concatenation (+) is used to join strings together.
* **Example**: If we run this line as is, it will output ahmed sami. If we change first\_name to "John" and last\_name to "Doe", it would output John Doe.

**Summary**

In this simple code snippet, we learned how to create and manipulate string variables in Python. We assigned values to variables, used comments for clarity, and printed the combined result to the console. Understanding these basics is essential for building more complex programs.

Extra python code for lesson 17

section 3

lesson 17

first\_name = "ahmed"

last\_name = 'sami'

# print(type(last\_name))

print(first\_name + " " + last\_name)

lesson 18

**Dr. Ahmed Sami's Explanation of the Code**

In this lesson, we will break down the provided code line by line. The purpose of this code is to demonstrate different ways to format and print strings in Python, specifically using variables for names and ages.

1. **name = "ahmed"**
   * **Usage**: This line initializes a variable named name and assigns it the string value "ahmed".
   * **Purpose**: It stores the name of a person that we will later use in our print statements.
2. **age = 27**
   * **Usage**: This line initializes a variable named age and assigns it the integer value 27.
   * **Purpose**: It stores the age of the person that we will use in our print statements.
3. **print("my name is ahmed I am 27 years old")**
   * **Usage**: This line prints the string directly to the console.
   * **Purpose**: It outputs a hardcoded message that introduces the name and age without using variables.
4. **# print("my name is ahmed I am 27 years old")**
   * **Usage**: This line is commented out (indicated by the #) and will not execute.
   * **Purpose**: It seems to be for reference or to demonstrate how the same message could be printed directly.
5. **print("my name is {name} I am {age} years old".format(name=name, age=age ))**
   * **Usage**: This line prints a formatted string using the format() method.
   * **Purpose**: It uses the variables name and age to dynamically insert their values into the string, allowing for a more flexible output.
6. **# print("my name is {name} I am {age} years old".format(name=name, age=age ))**
   * **Usage**: This line is also commented out and will not execute.
   * **Purpose**: Similar to the previous comment, it serves as a reference.
7. **print("my name is {my\_name} I am {my\_age} years old".format(my\_name=name, my\_age=age ))**
   * **Usage**: This line prints a formatted string using the format() method with different variable names (my\_name and my\_age).
   * **Purpose**: It demonstrates that we can use different names for the formatting variables while still utilizing the original variables name and age.
8. **name = "Mahmoud"**
   * **Usage**: This line reassigns the name variable to a new string value "Mahmoud".
   * **Purpose**: It changes the value of name to demonstrate that variables can be updated.
9. **age = 30**
   * **Usage**: This line reassigns the age variable to a new integer value 30.
   * **Purpose**: It changes the value of age to reflect a different age.
10. **print("my name is {my\_name} I am {my\_age} years old".format(my\_name=name, my\_age=age ))**
    * **Usage**: This line prints the updated values of name and age using the format() method.
    * **Purpose**: It shows the output with the new values assigned to the variables.
11. **# print("my name is {my\_name} I am {my\_age} years old".format(my\_name=name, my\_age=age ))**
    * **Usage**: This line is commented out and will not execute.
    * **Purpose**: Again, it serves as a reference.
12. **name = "Mahmoud"**
    * **Usage**: This line reassigns the name variable to "Mahmoud" again, but this time it will be used in the next print statement.
    * **Purpose**: It serves to demonstrate variable reassignment.
13. **age = 30**
    * **Usage**: This line reassigns the age variable to 30 again.
    * **Purpose**: Similar to before, it demonstrates variable reassignment.
14. **print(f"my name is {name} I am {age} years old")**
    * **Usage**: This line prints a formatted string using an f-string (formatted string literals).
    * **Purpose**: It provides a more concise and modern way to embed expressions directly within string literals.
15. **name = "Mahmoud Shaabo"**
    * **Usage**: This line reassigns the name variable to a new string value "Mahmoud Shaabo".
    * **Purpose**: It changes the value of name to demonstrate that variables can be updated again.
16. **age = 35**
    * **Usage**: This line reassigns the age variable to a new integer value 35.
    * **Purpose**: It updates the age of the person.
17. **# print("my name is {my\_name} I am {my\_age} years old".format(my\_name=name, my\_age=age ))**
    * **Usage**: This line is commented out and will not execute.
    * **Purpose**: It serves as a reference.
18. **print(f"my name is {name} I am {age} years old")**
    * **Usage**: This line prints the updated values of name and age again using an f-string.
    * **Purpose**: It shows the output with the most recent values assigned to the variables.

**Summary**: The provided code demonstrates different methods to format strings in Python, including hardcoded strings, the format() method, and f-strings. It illustrates how to use variables effectively to create dynamic outputs.

Section 3

Lesson 18

Extra python code

name = "ahmed"

age = 27

print("my name is ahmed I am 27 years old")

# print("my name is ahmed I am 27 years old")

print("my name is {name} I am {age} years old".format(name=name, age=age ))

# print("my name is {name} I am {age} years old".format(name=name, age=age ))

print("my name is {my\_name} I am {my\_age} years old".format(my\_name=name, my\_age=age ))

name = "Mahmoud"

age = 30

print("my name is {my\_name} I am {my\_age} years old".format(my\_name=name, my\_age=age ))

# print("my name is {my\_name} I am {my\_age} years old".format(my\_name=name, my\_age=age ))

name = "Mahmoud"

age = 30

print(f"my name is {name} I am {age} years old")

name = "Mahmoud Shaabo"

age = 35

# print("my name is {my\_name} I am {my\_age} years old".format(my\_name=name, my\_age=age ))

print(f"my name is {name} I am {age} years old")

**Ssection 3**

**Lesson 19**

In this lesson, we will go through the provided code line by line. The purpose of this code is to demonstrate various string methods available in Python that can be used to manipulate and analyze strings.

1. **name = "ahmed"**
   * **Usage**: This line initializes a variable named name and assigns it the string value "ahmed".
   * **Purpose**: It stores the name of a person that could be used later in the program, although it is not directly used in the subsequent lines.
2. **s = "hello world"**
   * **Usage**: This line initializes a variable named s and assigns it the string value "hello world".
   * **Purpose**: It sets up a sample string that will be used for demonstrating various string methods.
3. **# print(s)**
   * **Usage**: This line is commented out (indicated by the #) and will not execute.
   * **Purpose**: It seems to be for reference or to show how to print the string s, but it is not required for the current demonstration.
4. **# length**
   * **Usage**: This is a comment indicating that the following line will calculate the length of the string.
   * **Purpose**: It serves as a label for clarity.
5. **print(len(s))**
   * **Usage**: This line prints the length of the string s using the len() function.
   * **Purpose**: It shows how to determine the number of characters in the string, which in this case is 11.
6. **# capitalize**
   * **Usage**: This is a comment indicating that the following line will capitalize the first letter of the string.
   * **Purpose**: It serves as a label for clarity.
7. **print(s.capitalize())**
   * **Usage**: This line prints the string s with the first letter capitalized using the capitalize() method.
   * **Purpose**: It demonstrates how to format the string so that the first character is uppercase, resulting in "Hello world".
8. **# upper case**
   * **Usage**: This is a comment indicating that the following line will convert the string to uppercase.
   * **Purpose**: It serves as a label for clarity.
9. **print(s.upper())**
   * **Usage**: This line prints the string s in all uppercase using the upper() method.
   * **Purpose**: It demonstrates how to convert the entire string to uppercase, resulting in "HELLO WORLD".
10. **# lower case**
    * **Usage**: This is a comment indicating that the following line will convert the string to lowercase.
    * **Purpose**: It serves as a label for clarity.
11. **print(s.lower())**
    * **Usage**: This line prints the string s in all lowercase using the lower() method.
    * **Purpose**: It demonstrates how to convert the entire string to lowercase, resulting in "hello world".
12. **# swipe case**
    * **Usage**: This is a comment indicating that the following line will swap the case of the string.
    * **Purpose**: It serves as a label for clarity.
13. **print(s.swapcase())**
    * **Usage**: This line prints the string s with all uppercase letters converted to lowercase and vice versa using the swapcase() method.
    * **Purpose**: It demonstrates how to change the case of each character in the string, resulting in "HELLO WORLD".
14. **# replace**
    * **Usage**: This is a comment indicating that the following line will replace a substring.
    * **Purpose**: It serves as a label for clarity.
15. **print(s.replace("world", "everyone"))**
    * **Usage**: This line prints the string s with the substring "world" replaced by "everyone" using the replace() method.
    * **Purpose**: It demonstrates how to modify parts of the string, resulting in "hello everyone".
16. **# count character**
    * **Usage**: This is a comment indicating that the following line will count occurrences of a character.
    * **Purpose**: It serves as a label for clarity.
17. **char = "l"**
    * **Usage**: This line initializes a variable named char and assigns it the character "l".
    * **Purpose**: It sets up a character for counting its occurrences in the string s.
18. **print(s.count(char))**
    * **Usage**: This line prints the number of times the character stored in char appears in the string s using the count() method.
    * **Purpose**: It demonstrates how to count the occurrences of a specific character, which is 3 in this case.
19. **# startswith**
    * **Usage**: This is a comment indicating that the following line will check if the string starts with a specific substring.
    * **Purpose**: It serves as a label for clarity.
20. **print(s.startswith("he"))**
    * **Usage**: This line prints a boolean value indicating whether the string s starts with the substring "he" using the startswith() method.
    * **Purpose**: It demonstrates how to check the beginning of a string, which returns True.
21. **# endswith**
    * **Usage**: This is a comment indicating that the following line will check if the string ends with a specific substring.
    * **Purpose**: It serves as a label for clarity.
22. **print(s.endswith("s"))**
    * **Usage**: This line prints a boolean value indicating whether the string s ends with the substring "s" using the endswith() method.
    * **Purpose**: It demonstrates how to check the end of a string, which returns False.
23. **# split**
    * **Usage**: This is a comment indicating that the following line will split the string into a list of substrings.
    * **Purpose**: It serves as a label for clarity.
24. **print(s.split())**
    * **Usage**: This line prints a list of substrings obtained by splitting the string s using the split() method.
    * **Purpose**: It demonstrates how to separate a string into its individual words, resulting in ["hello", "world"].
25. **# find**
    * **Usage**: This is a comment indicating that the following line will find the position of a substring.
    * **Purpose**: It serves as a label for clarity.
26. **print(s.find("d"))**
    * **Usage**: This line prints the index of the first occurrence of the substring "d" in the string s using the find() method.
    * **Purpose**: It demonstrates how to locate a specific character within a string, returning the index 9.
27. **# Is all alphanumeric**
    * **Usage**: This is a comment indicating that the following line will check if the string is alphanumeric.
    * **Purpose**: It serves as a label for clarity.
28. **print(s.isalnum())**
    * **Usage**: This line prints a boolean value indicating whether the string s consists only of alphanumeric characters using the isalnum() method.
    * **Purpose**: It demonstrates how to check if a string contains only letters and numbers, which returns False in this case due to the space.
29. **# Is all alphabetic**
    * **Usage**: This is a comment indicating that the following line will check if the string is alphabetic.
    * **Purpose**: It serves as a label for clarity.
30. **print(s.isalpha())**
    * **Usage**: This line prints a boolean value indicating whether the string s consists only of alphabetic characters using the isalpha() method.
    * **Purpose**: It demonstrates how to check if a string contains only letters, which returns False in this case due to the space.
31. **# Is all numeric**
    * **Usage**: This is a comment indicating that the following line will check if the string is numeric.
    * **Purpose**: It serves as a label for clarity.
32. **print(s.isnumeric())**
    * **Usage**: This line prints a boolean value indicating whether the string s consists only of numeric characters using the isnumeric() method.
    * **Purpose**: It demonstrates how to check if a string contains only numbers, which returns False because the string contains letters.

**Summary**: The provided code demonstrates various methods available in Python for manipulating and analyzing strings. It showcases how to measure string length, change case, replace substrings, count characters, and check for specific conditions like whether a string starts or ends with certain substrings.

Extra python code for lesson 19

s = "hello world"

# print(s)

# length

print(len(s))

# capitalize

print(s.capitalize())

# upper case

print(s.upper())

# lower case

print(s.lower())

# swipe case

print(s.swapcase())

# replace

print(s.replace("world", "everyone"))

# count character

char = "l"

print(s.count(char))

# startswith

print(s.startswith("he"))

# endswith

print(s.endswith("s"))

# split

print(s.split())

# find

print(s.find("d"))

# Is all alphanumeric

print(s.isalnum())

# Is all alphabetic

print(s.isalpha())

# Is all numeric

print(s.isnumeric())