CHAP 2 : Data Structure هيكل البيانات

2-1-Variables: المتغيرات

In Python, variables are used to store data values. Variables do not need explicit declaration to reserve memory space. The declaration happens automatically when you assign a value to a variable. Variables can store different types of data, such as integers, floating-point numbers, strings, lists, and more.

1. Creating and Assigning Variables
2. Reassigning Variables
3. Multiple Variable Assignment
4. Swapping Variables
5. Global and Local Variables
6. Variable Types
7. Constants
8. Type Checking with type()
9. Deleting Variables
10. Dynamic Typing in Python
11. Variable Naming Rules
12. Variable Scope

Let's explore how variables work in Python with explanations and examples.

1. Creating and Assigning Variables انشاء و تعيين المتغيرات

You can create and assign a variable using the equals sign =. The value on the right side is assigned to the variable on the left side.

Example:

Python Code

```
# Assigning values to variables
x = 10
y = 3.14
name = "Ali"

# Printing variable values
print("x =", x)
print("y =", y)
print("name =", name)
```

Output:

```
x = 10
y = 3.14
name = Ali
```

2. Reassigning Variables إعادة تعيين المتغيرات

You can change the value of a variable by assigning a new value to it. Python is dynamically typed, so you can assign a different type to the same variable.

Example:

Python Code

```
# Reassigning variables
x = 100
print("x =", x)

# Changing the type of a variable
x = "Hello"
print("x =", x)
```

Output:

```
x = 100
x = Hello
```

3. Multiple Variable Assignment تعيينات متعددة للمتغيرات

You can assign values to multiple variables in a single line using multiple assignment.

Example:

Python Code

```
# Multiple assignment
a, b, c = 5, 10, 15

# Print the values
print("a =", a)
print("b =", b)
print("c =", c)
```

Output:

```
a = 5
b = 10
c = 15
```

4. Global and Local Variables المتغيرات العامة و المتغيرات المحلية

- **Local variables** are defined inside a function and are only accessible within that function.
- **Global variables** are defined outside of functions and are accessible throughout the entire program.

Example:

Python Code

```
# Global variable
x = "global"

def my_function():
    # Local variable
    x = "local"
```

```
    print("Inside function:", x)

# Calling the function
my_function()

# Outside the function
print("Outside function:", x)
```

Output:

```
Inside function: local
Outside function: global
```

Example with Global Keyword:

You can use the global keyword to modify a global variable inside a function.

Python Code

```
x = "global"

def my_function():
    global x
    x = "modified global"
    print("Inside function:", x)

# Calling the function
my_function()

# Outside the function
print("Outside function:", x)
```

Output:

```
Inside function: modified global
Outside function: modified global
```

5. Variable Types انواع المتغيرات

6.1. Integer Variables

python
Code

```
a = 10
b = -5
print("a =", a, "b =", b)
```

Output:

```
a = 10 b = -5
```

6.2. Float Variables

Python Code

```
x = 3.14
y = -0.001
print("x =", x, "y =", y)
```

Output:

```
x = 3.14 y = -0.001
```

6.3. String Variables

Python Code

```
first_name = "John"
last_name = "Doe"
full_name = first_name + " " + last_name
print("Full Name:", full_name)
```

Output:

Full Name: John Doe

6. Constants الثوابت

Python doesn't have a built-in constant type, but by convention, you can define constants using all capital letters. Constants are variables whose values should not be changed during the execution of the program.

Example:

Python Code

```
PI = 3.1416
GRAVITY = 9.8

print("PI =", PI)
print("Gravity =", GRAVITY)
```

Output:

PI = 3.1416
Gravity = 9.8

6. Type Checking with type() التحقق من طبيعة المتغير

You can check the data type of a variable using the type() function.

Example:

Python Code

```
x = 10
y = 3.14
name = "Alice"

print("Type of x:", type(x))
print("Type of y:", type(y))
print("Type of name:", type(name))
```

Output:

Type of x: <class 'int'>
Type of y: <class 'float'>
Type of name: <class 'str'>

7. Deleting Variables حذف المتغيرات

You can delete a variable using the del keyword.

Example:

Python Code

```
x = 10
print("x =", x)
```

```
# Deleting variable x
del x

# Trying to access the deleted variable will raise an error
# print(x)  # Uncommenting this line will raise a NameError
```

Output:

```
x = 10
```

8. Dynamic Typing in Python

Python is dynamically typed, meaning you don't need to declare a variable's type explicitly. The type of the variable is determined based on the value assigned to it.

Example:

Python Code

```
x = 10     # x is an integer
print("x =", x)

x = 3.14   # Now x is a float
print("x =", x)

x = "Hello"  # Now x is a string
print("x =", x)
```

Output:

```
x = 10
x = 3.14
x = Hello
```

9. Variable Naming Rules قواعد تسمية المتغيرات

There are certain rules and conventions to follow when naming variables:

- Variable names must start with a letter or an underscore _.
- The rest of the variable name can contain letters, numbers, or underscores.
- Variable names are case-sensitive (myVariable and myvariable are different).

Example of Valid and Invalid Variable Names:

```
# Valid variable names
age = 25
2name = " Alice "
My-variable = 10


# Invalid variable names (will cause syntax errors)
# 2name = "Alice"  # Cannot start with a number
# my-variable = 10  # Hyphens are not allowed
```

10-Display Variables: عرض المتغيرات

username=Alice

age=25

```
main.py
print(username + " is " + str(age) + " years old")

print(f"Hello, {username}")
print(f"{username} is {age} years old")

print("Hello, {}!".format(username))
```

replaced with value

```
Run  main
Hello, Alice
Alice is 25 years old
Hello, Alice
Alice is 25 years old
Hello, Alice!

Process finished with exit code 0
```

```
main.py
username = "Alice"
print(username)

print("Hello, " + username)
```

plus (+) Used for Concate

```
Alice

Run  main
C:\Users\Administrator\PycharmProjects\FormattingVariable\.venv\Scripts\python.exe C:\Users\Administrator\Pychar
Alice
Hello, Alice

Process finished with exit code 0
```

```
main.py
username = "Alice"
print(username)

print("Hello, " + username)

age = 25
print(username + " is " + str(age) + " years old")

print(f"Hello, {username}")
print(f"{username} is {age} years old")
```

Alice 25

```
Run  main
Alice
Hello, Alice
Alice is 25 years old
Hello, Alice
Alice is 25 years old

Process finished with exit code 0
```

main.py

```python
username = "Alice"
print(username)

print("Hello, " + username)

age = 25
print(username + " is " + str(age) + " years old")
```

Run main

```
Alice
Hello, Alice
Alice is 25 years old

Process finished with exit code 0
```

main.py

```python
username = "Alice"
print(username)

print("Hello, " + username)

age = 25
print(username + "is" + str(age) + "years old")
```

Alice 25