Lecture 4: Variables in Python1 (String Variables)

Just as a social researcher needs to classify phenomena and identify their characteristics, a programmer needs tools that enable them to represent reality within a program. A variable is the element that allows information to be stored and modified, while variable types serve as templates that define the nature of this information: is it a number? a string? a Boolean value? Through these fundamental concepts, the student begins to build a practical understanding of programming, where ideas are transformed into symbols, and symbols into executable instructions. These variables are not only essential for understanding the Python language, but also constitute an entry point to understanding how social phenomena are translated into data that can be analyzed and processed.

First: Introduction to Variables

1. **What is a variable?**

A variable is a name used to store a value inside a program. This value can be changed later, which is why it is called a "variable."

In social sciences, for example, a variable can be something like *age*, *gender*, *occupation*, or *political opinion* in a survey.

Example in Python:

```
nom = "Ahmed"
age = 30
profession = "enseignant"
```

2. Why do we use variables?

- To temporarily store data while the program is running.
- To make it easier to reuse values.
- To give information meaningful names.

3. Basic variable types in Python

Example	Description	Type
age = 25	Integer number	int
temperature = 22.5	Decimal number	float
name = "Fatima"	Text string	str
is_student = True	Boolean value	bool
opinions = ["yes", "no", "maybe"]	List of values	list
person = {"name": "Ali", "age": 40}	Dictionary (key-value pairs)	dict

4. Rules for naming variables in Python

For a variable name to be valid, it must follow these rules:

- Must begin with a letter or an underscore _ (underscore is typed with **AltGr+8** on the keyboard).
- Cannot contain spaces.
- Cannot start with a number.
- Should preferably be descriptive of its content.

Incorrect examples:

```
2name = "Ali"     # Wrong

user name = "Sara" # Wrong
```

Correct examples:

```
user_name = "Sara"   # Correct

age2 = 45
```

Note: In Python, the symbol # is used to write a **comment**.

A comment is text written inside the program but not executed. The interpreter completely ignores it. Comments are used to:

- Explain the code.
- Clarify the function of a specific part.
- Leave notes for other programmers (or for yourself in the future).

Example:

```
# This variable stores the user's age
age = 25
```

Second: String Variables in Python

1. Initial Definition

A string variable is a type of variable used to store a sequence of characters, such as a word, a sentence, or even a whole paragraph. In Python, this type is referred to as str, short for *string*. This means that when you declare a string variable, Python internally stores it as an object of type str.

```
message = "Welcome to the world of Python"
```

This example stores a sentence inside a variable named message.

2. Why do we need string variables?

Strings are an essential part of any interactive program. We use them to:

- Display messages to the user
- Input textual data such as names or email addresses

- Process sentences and words in applications like translation or search
- Create interactive interfaces based on language

3. How are strings written in Python?

Strings are written between quotation marks:

- Single quotes 'text'
- Double quotes "text"

Both are correct:

```
title = "Programming"
author = 'Ahmed'
```

Incorrect examples (mismatched quotes):

```
title = "Programming'
author = 'Ahmed"
```

4. Declaring a string variable

Declaring a variable in any programming language means informing the interpreter that you will use a certain variable and specifying its type (integer, string, etc.). This helps the program allocate memory space for it.

name = "Ahmed"

greeting = "Welcome"

A string can contain letters, numbers, symbols, and even spaces.

5. Is text limited to letters only?

No, a string can include:

- Arabic or Latin letters
- Numbers: "123"
- Symbols: "@#&*"
- Spaces: " " or "Badreddine"

All of these are considered strings as long as they are enclosed in quotation marks.

6. Difference between text and numeric values

```
age = 20       # Integer
age_text = "20" # String representing a number
```

Although they look similar, age can be used in mathematical operations, while age_text is treated purely as text.

7. Converting values to strings

You can convert any value into a string using the str() function:

```
score = 95

text_score = str(score)

print("Your score is " + text_score)
```

 Note: After converting a numeric value to a string, it cannot be used in arithmetic operations.

```
score = 95

text_score = str(score)

print("Your score is " + text_score)

somme = text_score + 2   # Error!

print(somme)
```

Python will raise an error because text_score is of type str.

8. Key string functions in Python

In programming, functions are essential tools for organizing code and simplifying tasks. A function is simply a ready-made recipe to perform a specific task, which can be called whenever needed instead of rewriting the same instructions repeatedly.

Functions are always expressed by their name followed by parentheses:

function_name()

One of the most common functions is print().

Since strings are among the most frequently used data types in programming, Python provides a rich set of built-in functions to process them. These functions help us:

- Modify strings
- Search within them
- Split or join them
- Convert between uppercase and lowercase
- And many other vital tasks

A. Basic Modification Functions

Function	Purpose	Example	Result
upper()	Convert text to uppercase	magis = "hello".upper() print(magis)	"HELLO"
lower()	Convert text to lowercase	minis = "HELLO".lower() print(minis)	"hello"
strip()	Remove spaces from start and end	pasvid = " text ".strip() print(pasvid)	"text"
replace(old, new)	Replace part of the text	remp = "abc".replace("a", "z") print(remp)	"zbc"

Combined example:

```
variable = "   HELLO WORLD   ".replace("WORLD", "PYTHON").strip().lower()
print(variable)  # Output: hello python
```

B. Search and Validation Functions

Function	Purpose	Example	Result
find(sub)	Find first position of substring (index starts at 0)	first = "hello".find("l") print(first)	2
count(sub)	Count occurrences of substring	npart = "banana".count("a") print(npart)	3
startswith(prefix)	Check if text starts with prefix	deb = "hello".startswith("he") print(deb)	True
endswith(suffix)	Check if text ends with suffix	fin = "hello".endswith("lo") print(fin)	True

C. Splitting and Joining Functions

Function	Purpose	Example	Result
split(sep)	Split text into a list	text = "Ahmed,Sara,Youssef,Laila" names = text.split(",") print(names)	['Ahmed', 'Sara', 'Youssef', 'Laila']
join(list)	Join list into text	names = ['Ahmed', 'Sara', 'Youssef', 'Laila'] text = " and ".join(names) print(text)	"Ahmed and Sara and Youssef and Laila"

D. Content-Type Checking Functions

Function	Purpose	Example	Result
isalpha()	Check if all characters are alphabetic	alpha = "abc".isalpha() print(alpha)	True
isdigit()	Check if all characters are digits	numeri = "123".isdigit() print(numeri)	True
isalnum()	Check if all characters are alphanumeric	alphanum = "abc123".isalnum() print(alphanum)	True

9. Inputting Text from the User

In programming, we often need the user to enter data while the program is running, such as their name, age, or any other information. This process is called **user input**, and it is the way a program receives data directly from the person using it.

In Python, we use the input() function to get input from the user. This function pauses the program temporarily and waits for the user to type a value from the keyboard, then returns this value as a string (str).

```
name = input("What is your name? ")
print("Welcome, " + name)
```

Explanation:

- The input("What is your name? ") function asks the user to enter their name.
- The entered value is stored in the variable name.
- Then we use print() to display a greeting message using the name provided.

Important notes:

- Everything entered via input() is considered text (even if it looks like a number).
- If you want to convert the input into a number, use int() or float().

Example:

```
age = int(input("How old are you? "))
```

10. Operations on Strings

- **Concatenation (joining strings):**

```
first_name = "Ahmed"
last_name = "Hamdi"
full_name = first_name + " " + last_name
print(full_name)  # Output: Ahmed Hamdi
```

- **Repetition:**

```
greeting = "Hello "
print(greeting * 3)  # Output: Hello Hello Hello
```

- **Length:**

```
text = "University of Algiers"
print(len(text))  # Output: 19
```

11. String Checking

```
text = "Programming is fun"
print("fun" in text)         # True
print(text.startswith("Pro")) # True
print(text.endswith("n"))     # True
```

Practical Exercise

```
name = input("Enter your name: ")
greeting = "Welcome, " + name
print(greeting.upper())
```

Task: Modify the greeting or include the last name as well.

Model Solution:

```
# Input first and last name from the user
first_name = input("Enter your first name: ")
last_name = input("Enter your last name: ")
# Create greeting using full name
greeting = "Welcome, " + first_name + " " + last_name
# Display greeting in uppercase
print(greeting.upper())
```

Notes:

- You can adjust the format to include different welcoming phrases such as: "Welcome to the course, ..." or "Greetings to Professor ..."
- You can use .title() instead of .upper() for a more elegant format:

```
print(greeting.title())  # Output: Welcome, Ahmed Ben Youssef
```