

Lec : (2)

Fundamentals of Programming in Python

Lecture Objectives

By the end of this lecture, the student will be able to:

- Understand the concept of programming and logic.
- Write a simple code using Python.
- Work with variables, conditions, loops, and functions.

Part 1: Quick Introduction

What is Programming?

Programming is a way to make the computer execute commands step by step through code written in a language that the machine can understand.

Why Python?

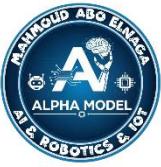
- Easy to read and understand.
- Used in many fields (Artificial Intelligence, Web Development, Data Analysis, Robotics).
- Large community and strong support.

Part 2: Working Environment

- Install Python from the official website: [python.org](https://www.python.org)
- Or use **Google Colab** directly without installation.

Writing Your First Python Code

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



Explanation:

- **print()** is a **built-in function** in Python used to display text or values on the screen.
- The text inside the parentheses ("Hello, world!") is what will be shown when the program runs.

Output:

Hello, world!

💡 Part 3: Variables and Data Types

Variables

Variables store values in the computer's memory:

```
name = "Mahmoud"  
age = 20  
height = 1.75
```

Rules:

- The name **cannot start with a number**.
- **No spaces** are allowed (use an underscore _ instead).
- **Case sensitive** → Name ≠ name.

Data Types in Python



Type	Example	Description
int	5	Integer (whole number)
float	3.14	Decimal (floating-point number)
str	"Hello"	String (text)
bool	True / False	Boolean value (logical true or false)
list	[1, 2, 3]	List (ordered collection of items)
dict	{"name": "Ali", "age": 18}	Dictionary (key-value pairs)

Part 4: Operators

Arithmetic Operators

```
x = 10
y = 3

print(x + y)      # Addition
print(x - y)      # Subtraction
print(x * y)      # Multiplication
print(x / y)       # Division
print(x // y)     # Floor Division (integer result)
print(x % y)       # Modulus (remainder)
print(x ** y)      # Exponentiation (power)
```

Logical (Comparison) Operators

```
a = 10
b = 5

print(a > b)      # True → greater than
print(a == b)      # False → equal to
print(a != b)      # True → not equal to
```



□ Part 5: Conditional Statements (If–Else)

```
age = int(input("Enter your age: "))

if age >= 18:
    print("You are an adult")
else:
    print("You are a minor")
```

Explanation:

- The program takes the user's age as input.
- If the age is **18 or more**, it prints "*You are an adult*".
- Otherwise, it prints "*You are a minor*".

. الأخطاء مش نهاية الطريق، دي بداية الفهم

Combined Condition

```
if age >= 18 and age <= 60:
    print("Working age")
```

Explanation:

The condition checks **two requirements** using and:

- Age is **18 or older, and**
- Age is **60 or younger**.

If both are true → it prints "*Working age*".

Part 6: Loops



for loop

```
for i in range(5):
    print("Iteration:", i)
```

Explanation:

The loop repeats the code **5 times** (from 0 to 4).

Each time, the variable **i** takes the next number in the range.

while loop

```
count = 0
while count < 3:
    print("Count is:", count)
    count += 1
```

Explanation:

The code runs **as long as** the condition **count < 3** is true.

Each iteration increases **count** by 1.

Part 7: Functions

```
def greet(name):
    print("Hello", name)

greet("Mahmoud")
```

Explanation:

A **function** is a reusable block of code.

Here, **greet()** takes one parameter **name** and prints a message.

Function with Return Value

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

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



Explanation:

The add() function returns the sum of a and b.

The returned value is stored in result and then printed.

اتعلم تصبر على الكود، زيه زي الحياة تحتاج وقت ⏱

Part 8: Lists

```
fruits = ["apple", "banana", "cherry"]
print(fruits[0])    # apple

fruits.append("orange")
print(fruits)
```

Explanation:

- A **list** is a collection of items stored in one variable.
- You can access elements by their **index** (starting from 0).
- The append() function adds a new item to the end of the list.

Looping Through a List

```
for fruit in fruits:
    print(fruit)
```

Explanation:

The loop goes through each item in the list and prints it.

□ Part 9: Dictionaries

```
person = {"name": "Ali", "age": 22, "city": "Cairo"}
print(person["name"])
person["age"] = 23
```

Explanation:



- A **dictionary** stores data in **key–value pairs**.
- You can access a value using its key (e.g., person["name"]).
- Values can be updated by assigning a new one.

Part 10: Mini Project

Simple Student Registration Program

```
students = []

for i in range(3):
    name = input("Enter name: ")
    age = int(input("Enter age: "))
    students.append({"name": name, "age": age})

for s in students:
    print(s["name"], "-", s["age"])
```

Explanation:

- The program collects names and ages of 3 students.
- Each student is stored as a dictionary inside the list students.
- At the end, it prints each student's name and age.

"**وَقُلْ رَبِّ زَنْبِي عَلِمًا**" 

النهاية