Introduction to

Python Programming

الوحدة 03: من الخوارزمية الى برنامج بسيط

ما هي الخوازمية؟

الخوارزمية هي مجموعة من الخطوات المتسلسلة المصممة لحل مشكلة معينة أو لتنفيذ مهمة محددة.

1

منتهية

لها بداية ونهاية واضحتان — حيث تنتهي بعد عدد محدد من الخطوات.

2

دقيقة

كل خطوة واضحة وغير غامضة، ومحددة بدقة دون أي مجال لسوء الفهم.

3

فعّالة

كل خطوة قابلة للتنفيذ وتؤدي إلى نتيجة صحيحة للمشكلة المعطاة.

From Algorithm to Program

Program: مجموعة من التعليمات مكتوبة بلغة برمجة يمكن للحاسوب تنفيذها لأداء مهمة محددة.

Programming: عملية تحويل الخوارزمية إلى برنامج قابل للتنفيذ باستخدام لغة برمجة معينة.

Programming Language: مجموعة من الكلمات والرموز التي توفر طريقة للتعبير عن التعليمات وهياكل البيانات مثل Python, java, C++

Programming Languages

Python

Since 1991

AI, Data Science, Web

Java

Since 1995

Enterprise, Android

C / C++

Since 1972

Systems, Gaming

JavaScript

Since 1995

Web, Frontend, Backend

C#

Since 2000

Games (Unity), Windows

PHP

Since 1994

Web Backend, CMS

Ruby

Since 1995

Web, Scripting

Swift

Since 2014

iOS / macOS Apps

Why Python is the Most Used Language Today

Python

- ✓ سهلة التعلم — قريب من اللغة الطبيعية.
- ✓ عالية المستوى (High-Level) — يُخفي التفاصيل المعقدة مثل إدارة الذاكرة.
- ✓ قابلة للنقل بين الأنظمة(Portable) — يمكن تشغيل نفس الكود على ويندوز، ماك، ولينكس.
- ✓ مكتبات واسعة (Extensive Libraries) — مجموعة ضخمة من الحزم المدمجة والخارجية.
- ✓ تحديد نوع المتغيرات ديناميكيًا (Dynamic Typing) — لا حاجة لإعلان نوع المتغيرات

Example: Sum of Three Integers

Algorithm (Pseudocode)

```
Algorithm sum;
Var Nb1,Nb2,Nb3, Result : integer;
begin
  write("Enter three numbers");
  read(Nb1);
  read(Nb2);
  read(Nb3);
  Result := Nb1 + Nb2 + Nb3;
  write(Result);
end
```

Python Program

```
print("Enter three numbers")
Nb1 = int(input())
Nb2 = int(input())
Nb3 = int(input())
Result = Nb1 + Nb2 + Nb3


print(Result)
```

Python Syntax Rules

قواعد كتابة بايثون

ثلاث قواعد يجب أن يعرفها كل مبرمج بايثون منذ اليوم الأول:

① Indentation Matters

Python uses spaces/tabs to define code blocks
Wrong indentation = error.

```
# ✅ Correct
if 5 > 3:
    print("yes")

# ❌ Wrong
if 5 > 3:
print("yes")
```

② Comments with

Lines starting with # are ignored by Python — use them to explain your code.

```
# This is a comment
x = 10  # inline comment
```

③ Case Sensitivity

Python is case-sensitive: Name and name are two different variables!

```
name = "Alice"   # different
Name = "Bob"    # variables!
```

Where Do You Write Python?

أين تكتب بايثون؟

You need an editor or environment to write and run Python code:

IDLE

Built-in

مرفق مع بايثون. مثالي للمبتدئين — لا حاجة لتثبيت إضافي خارج بايثون نفسه.

- Free & pre-installed
- Simple interface

VS Code

Recommended

Visual Studio Codeمحرر قوي ومجاني من مايكروسوفت. يستخدمه الملايين من المحترفين حول العالم.

- Auto-complete & hints
- Extensions ecosystem

Online Editors

No Install

تشغيل بايثون مباشرة في المتصفح. ممتاز للتجارب السريعة والتعلم دون أي إعداد.

- replit.com
- trinket.io
- python.org/shell

Your First Python Program — "Hello, World!"

Why Hello World?

هو البرنامج الأول التقليدي في كل لغة برمجة. وهو يُستخدم للتحقق من أن بيئة العمل لديك مُعدّة بشكل صحيح وأنك قادر على تشغيل البرنامج بنجاح..

hello.py

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

OUTPUT

```
Hello, World!
```

Variables & Naming Rules

المتغيرات وقواعد التسمية

المتغير هو موقع في الذاكرة يحمل اسماً ويُستخدم لتخزين البيانات.

Examples

```
# Valid names
age = 25
_score = 98.5
first_name = "Alice"

# Invalid , would cause error
# 1score = 10  ← starts with digit
# my score = 5 ← has a space
```

Naming Rules

✅يمكن أن يبدأ بحرف أو _

✅يمكن أن يحتوي على حروف، أرقام، و _

❌لا يمكن أن يبدأ برقم

❌لا يمكن أن يحتوي على مسافات

❌لا يمكن أن يكون كلمة محجوزة (مثل: if، for...)

Types of Variables

أنواع المتغيرات

int

Integer

```
x = 5
y = -3
z = 120
```

Whole numbers
Size: 2 or 4 bytes

float

Real Number

```
pi = 3.14
g = -9.8
tax = 0.2
```

Decimal numbers
Size: 4 or 8 bytes

bool

Boolean

```
is_on = True
has_id = False
```

True / False only
Size: 1 bit

str

String

```
name = "Alice"
msg = 'Hi!'
```

Text in quotes
Size: varies

Variables

المتغيرات

يتم تعريف المتغير باستخدام عملية الإسناد =

بايثون لغة ذات نوع ديناميكي(dynamically typed):

- لا حاجة لتصريح نوع المتغير مسبقًا.
- يمكن أن يتغير نوع المتغير أثناء تنفيذ البرنامج.

يمكن أن تأتي القيم من: ثوابت، عمليات حسابية بين متغيرات أخرى، أو نتيجة دالة.

The print() Function

```
# Print the str “Hello world!”
print(“Hello, World!")

# Print the int 10
print(10)

# Print the value of the variable name
name = “Ali”
print(name)

# Print a str and a variable
age = 20
print(“Age: ", age)
```

```
Output

Hello, World!
10
Ali
Age: 20
```

- **print()** تُستخدم في لغة بايثون لعرض (إظهار) النصوص أو القيم على الشاشة
- يمكنك استخدامها لعرض **نص، رقم،** أو حتى **قيمة متغير.**

The input() Function

```
# Read user input and convert

name = input()
name = input("Your name: ")
# Convert to integer
age = int(input("Your age: "))

# Convert to float
gpa = float(input("Your GPA: "))
```

```
▶ Output

Your name: Alice
Your age: 20
Your GPA: 3.8
```

① يقرأ البيانات التي يُدخلها المستخدم من لوحة المفاتيح.

② يُرجع دائمًا نصًا (str)

③ استعمل int() لتحويله إلى عدد صحيح.

④ استعمل float()لتحويله إلى عدد عشري.

⑤ يمكن عرض رسالة توجيه (prompt) داخل الأقواس.

Two Ways to Use input()

Style 1 — Separate print()

```
print("Enter Nb1:")
Nb1 = int(input())
print("Enter Nb2:")
Nb2 = int(input())
```

يستعمل دالة print() منفصلة قبل كل input() يكون أكثر تفصيلاً، لكنه يفصل بوضوح بين الرسالة وقراءة البيانات.

Style 2 — Inline Prompt ✓ Preferred

```
Nb1 = int(input("Enter Nb1: "))
Nb2 = int(input("Enter Nb2: "))
```

يتم تمرير الرسالة مباشرة داخل input() هذا الأسلوب أقصر وأكثر اختصارًا , وهو الأسلوب المفضل في بايثون.

Arithmetic Operators

العمليات الرياضية

Operator	Description	Example	Result
+	Addition	5 + 3	8
-	Subtraction	10 – 4	6
*	Multiplication	6 * 3	18
/	Division (returns float)	8 / 2	4.0
//	Floor Division (DIV)	7 // 2	3
%	Modulus (MOD)	7 % 2	1
**	Exponentiation (power)	3 ** 2	9

Arithmetic — Code & Output

```
# Code
a = 10
b = 3

print("Addition:",    a + b)
print("Subtraction:", a - b)
print("Multiply:",    a * b)
print("Division:",    a / b)
print("Floor Div:",   a // b)
print("Remainder:",   a % b)
print("Power:",       a ** b)
```

```
▶ Run Output

Addition:    13

Subtraction: 7

Multiply:    30

Division:    3.333...

Floor Div:   3

Remainder:   1

Power:       1000
```

Comparison Operators

عمليات المقارنة

Operator	Description	Example	Result
==	Equal to	5==5	True
!=	Not equal to	5 != 3	True
<	Less than	7<8	True
>	Greater than	1>2	False
<=	Less or equal than	5<=8	True
>=	Greater or equal than	5>=8	False

Exercise : Seconds Converter

اقرأ مدة زمنية بالثواني، ثم اعرضها على الشكل: $h : m : s$

```
Seconds = int(input("Enter seconds: "))

# Hours
Nh = Seconds // 3600

# Minutes (remaining after hours)
Nm = (Seconds % 3600) // 60

# Seconds remainder
Ns = Seconds % 60

print(Nh, ":", Nm, ":", Ns)
```

Example Run

```
Input:  3750 seconds
Output:  1 : 2 : 30
```

Key Operators

`//` Floor division → whole part
`%` Modulus → remainder
`3600` = 60 × 60 (seconds in 1 hr)

Strings in Python

```
text = "Hello, Python!"

print(text)
```

Definition of String

السلسلة النصية (String) هي تسلسل من الCharacters محاط بعلامات اقتباس مفردة أو مزدوجة. تُستخدم السلاسل النصية لتخزين البيانات النصية في بايثون، وتدعم العديد من العمليات المدمجة.

Single quotes

```
name = 'Sara'
```

Double quotes

```
name = "Sara"
```

Print it

```
print(name)   # Sara
```

Concatenation & Repetition

Concatenation (+)

يقوم بربط السلاسل النصية باستخدام الرمز +.

```
first_name = "Sara"
family_name = "Ibrahimi"
full_name = first_name + " " +
family_name
print(full_name)
# Output: Sara Ibrahimi
```

Repetition (*)

ويقوم بتكرار السلسلة النصية عدة مرات باستخدام الرمز *.

```
first_name = "Sara"
print(first_name * 3)

# Output: SaraSaraSara
```

len() Function & Indexing

len() — String Length

تحسب عدد الcharacters في سلسلة نصية.

```
print(len("Sara Ibrahimi"))  # 12
```

Indexing — Access Characters

كل محرف له index يبدأ من 0

```
name[0] → "S"  |  name[1] → "a"
```

"Sara" Index Map

Last index = len(string) - 1

find() · replace() · count()

```
text = "Programming with Python"
```

find()

تُرجع موضع (index) أول تطابق لسلسلة فرعية داخل النص

```
text.find("Python")   # → 17
```

replace()

تستبدل سلسلة فرعية بأخرى، وتُرجع سلسلة جديدة.

```
text.replace("Python", "Java")   # →
"Programming with Java"
```

count()

تقوم بعدّ عدد مرات ظهور محرف أو سلسلة فرعية داخل النص.

```
text.count("m")   # → 2
```

upper() · lower() · title()

```
text = "Programming with Python"
```

upper()

تحوّل جميع المحارف إلى UPPERCASE

Output:

```
"PROGRAMMING WITH PYTHON"
```

lower()

تحوّل جميع المحارف إلى lowercase

Output:

```
"programming with python"
```

title()

Capitalizes the first letter of each word

Output:

```
"Programming With Python"
```

Membership Operators: in & not in

يتحقق مما إذا كانت سلسلة فرعية موجودة (أو غير موجودة) داخل سلسلة نصية أخرى.

```
text = "Programming with Python"
```

in operator

Returns True if the substring IS found.

```
"Python" in text
# → True
```

not in operator

Returns True if the substring is NOT found.

```
"Java" not in text
# → True
```

Quick Reference: String Methods

`+`	**Concatenation**	Joins two strings
`*`	**Repetition**	Repeats a string N times
`len()`	**Length**	Number of characters
`[i]`	**Indexing**	Access character at index i
`find()`	**Search**	Index of first match
`replace()`	**Replace**	Swap one substring for another
`count()`	**Count**	Occurrences of a substring
`upper()`	**Case**	ALL UPPERCASE
`lower()`	**Case**	all lowercase
`title()`	**Case**	Each Word Capitalized
`in / not in`	**Membership**	Check if substring exists