

Lec-3

User defined function

- ❑ Exceptions and error handling (try, except)
- ❑ File handling: opening, reading, writing, closing
- ❑ Working with text files and CSV files
- ❑ Using the os library for file and folder management

1. (1-5) أسئلة سريعة عن lists & dicts & functions.

2. (6-10) دالة تحسب Factorial مثال سريع: باستخدام loop.

3. (11-15) إعلان موضوع اليوم: "إزاي نتعامل مع الملفات ونمنع الأخطاء".

مراجعة + مستوى متقدم — الدقيقة 40-16: User-defined Functions

4. (16-20)تعريف عام function = block of reusable code.

5. (21-25) بارامترات متعددة + return:

ما هي الـ Function ؟

مجموعة أوامر (كود) بتنكتب مرة وتنستخدم أكثر من مرة: **الدالة (Function)**.

• بتساعد على:

1. (Reusability) إعادة الاستخدام.

2. (Modularity) تنظيم الكود.

3. (Maintenance) تسهيل الصيانة.

4. (Less repetition → Less bugs) تقليل الأخطاء.

◆ 1. إنشاء Function

Copy code □

python

```
def greet():
    print("Hello, World!")

greet()    # استدعاء الدالة
```

2. Parameters (معاملات)

الدالة ممكن تأخذ مدخلات.

Copy code □

python

```
def greet(name):
    print(f"Hello, {name}!")

greet("Ahmed")
```

◆ 3. Return Values (القيم المرجعة)

الدالة ممكن ترجع قيمة

Copy code □

python

```
def add(x, y):
    return x + y

result = add(5, 3)
print(result)    # 8
```

4. Default Parameters (قيم افتراضية)

Copy code

python

```
def greet(name="Guest"):
    print(f"Welcome, {name}!")

greet()          # Welcome, Guest
greet("Mona")    # Welcome, Mona
```

5. Keyword Arguments

ممكن نحدد اسم البراميتر.

Copy code

python

```
def student_info(name, age):
    print(f"Name: {name}, Age: {age}")

student_info(age=22, name="Ahmed")
```

◆ عدد غير محدد من البراميترز . 6

*args → Tuple من القيم

Copy code

python

```
def add_all(*nums):
    return sum(nums)

print(add_all(1, 2, 3, 4))    # 10
```

من القيم **kwargs** → Dictionary

Copy code

python

```
def show_info(**info):
    for key, value in info.items():
        print(f"{key}: {value}")

show_info(name="Ahmed", age=22, city="Cairo")
```

◆ 7. Scope (نطاق المتغيرات)

متغير جوة الدالة: **Local**.

متغير خارج كل الدوال: **Global**.

• **Local:** متغير حارج كل الدوال.

Copy code

python

```
x = 10    # global

def test():
    x = 5    # local
    print("Inside:", x)

test()
print("Outside:", x)
```

استخدام `global` جوة دالة

Copy code

python

```
count = 0

def increase():
    global count
    count += 1

increase()
print(count) # 1
```



8. توثيق الدوال (Docstring) ◆

Copy code

python

```
def add(x, y):
    """This function returns the sum of two numbers"""
    return x + y

print(add.__doc__)
```

(دوال مختصرة) Lambda Functions .9 ◆

- دوال بدون اسم، مناسبة للعمليات الصغيرة.

Copy code

python

```
square = lambda x: x**2
print(square(5)) # 25
```

◆ 10. أمثلة واقعية

دالة للتحقق من باسورد:

Copy code

python

```
def check_password(pw):  
    if pw == "1234":  
        return True  
    return False
```

دالة لإرجاع أكبر رقم:

Copy code

python

```
def find_max(numbers):  
    return max(numbers)  
  
print(find_max([3, 9, 1, 7]))
```

◆ ما هو File Handling؟

بيانون بيوفر أدوات للتعامل مع الملفات

- فتح ملف (open)
- قراءة / كتابة / تعديل
- إغلاق الملف (close)



1. فتح ملف - open()

A screenshot of a dark-themed code editor. In the top right corner, it says "python". On the left, there is a "Copy code" button with a clipboard icon. The main area contains the following Python code:

```
file = open("test.txt", "mode")
```

أوضاع الفتح (Modes):

- قراءة (افتراضي) – خطأ لو الملف مش موجود → "r" .
- كتابة (يمسح الملف القديم) – ينشئ ملف لو مش موجود → "w" .
 - يكتب في آخر الملف بدون مسح – (Append) إضافة → "a"
 - إنشاء ملف جديد – خطأ لو الملف موجود → "x" .
- نص (افتراضي) → "t" مع .
- ثنائي → "b" مع (binary)

أوضاع متقدمة . 6

- قراءة وكتابة (لا ينشئ جديد) → "r+" .
- كتابة + قراءة (يمسح القديم) → "w+" .
 - إضافة + قراءة → "a+" .

♦ 2. الكتابة في الملفات

Copy code

python

```
f = open("data.txt", "w")
f.write("First Line\n")
f.write("Second Line\n")
f.close()
```

استخدام `()writelines`

Copy code

python

```
lines = ["One\n", "Two\n", "Three\n"]
f = open("data.txt", "w")
f.writelines(lines)
f.close()
```



♦ 3. القراءة من الملفات

Copy code

python

```
f = open("data.txt", "r")

print(f.read())          # يقرأ كل الملف
print(f.read(5))         # يقرأ أول 5 حروف
print(f.readline())       # يقرأ سطر واحد
print(f.readlines())      # فيها كل السطور List يرجع

f.close()
```

(♦ 4. استخدام `with` (أفضل طريقة)

- يفتح ويغلق الملف تلقائياً.

Copy code

python

```
with open("data.txt", "r") as f:
    content = f.read()
    print(content)
```

◆ 5. التعامل مع الملفات الثنائية (Binary)

Copy code

python

```
with open("image.jpg", "rb") as f:  
    data = f.read()  
  
with open("copy.jpg", "wb") as f:  
    f.write(data)
```

◆ 7. مكتبة os مع الملفات

Copy code

python

```
import os  
  
print(os.getcwd())          # معرفة مكانك الحالي  
os.rename("data.txt", "new.txt") # إعادة تسمية  
os.remove("new.txt")         # حذف ملف  
os.mkdir("myfolder")         # إنشاء 폴در  
os.rmdir("myfolder")        # حذف 폴در (فاضي)
```

◆ 8. مكتبة shutil (لنسخ ونقل)

Copy code

python

```
import shutil  
  
shutil.copy("data.txt", "copy.txt")   # نسخ ملف  
shutil.move("copy.txt", "backup/")    # نقل ملف
```

◆ 9. التعامل مع CSV (كمثال عملي)

Copy code

python

```
import csv

# كدك CSV
with open("data.csv", "w", newline="") as f:
    writer = csv.writer(f)
    writer.writerow(["Name", "Age"])
    writer.writerow(["Ahmed", 22])
    writer.writerow(["Mona", 21])

# قراءة CSV
with open("data.csv", "r") as f:
    reader = csv.reader(f)
    for row in reader:
        print(row)
```



◆ إيه هي الـ Exceptions ؟

- الـ **Exception** (Runtime Error). معناها خطأ يحصل أثناء تشغيل البرنامج.
 - لو مش عالجه → البرنامج هيف ويشهد رسالة خطأ.
- إنك تتحكم في الأخطاء وما تخليش البرنامج ينهار الهدف من **Error Handling**.

♦ بناء جملة الـ Error Handling

في try – except – else – finally Python نستخدم

Copy code

python

```
try:  
    الكود اللي ممكن يعمل خطأ #  
    num = int(input("Enter a number: "))  
    result = 10 / num  
except ZeroDivisionError:  
    print("X Error: You can't divide by zero.")  
except ValueError:  
    print("X Error: Please enter a valid number.")  
else:  
    print("✓ Result:", result)  
finally:  
    print("✓ Done (this block always runs.)")
```

لشرح:

• تكتب الكود المتوقع يعمل مشكلة → **try**.

• تكتب إيه يحصل لو الخطأ حصل → **except**.

• يتنفذ لو مفيش أي خطأ → **else**.

• يتنفذ في جميع الأحوال (مفید لغلق الملفات أو الاتصال) → **finally**.

♦ استخدام Exception عام

لو مش عارف نوع الخطأ، تقدر تستخدم:

Copy code

python

```
try:  
    x = 10 / 0  
except Exception as e:  
    print("Error happened:", e)
```

(إنشاء خطأ بنفسك) Raise Exception ◆

ممكن تطلق خطأ لو شرط معين مش متحقق:

Copy code

python

```
age = -5
if age < 0:
    raise ValueError("Age can't be negative")
```

Practical Real-Life Example ◆

Copy code

python

```
def read_file(filename):
    try:
        with open(filename, "r") as file:
            return file.read()
    except FileNotFoundError:
        return "✗ File not found."
    except PermissionError:
        return "✗ You don't have permission to open this file."
    finally:
        print("⌚ File handling attempt finished.")
```

```

# Menu (Control Flow + Loops)
# -----
def menu():
    students = load_students()
    while True:
        print("\n===== Student Management System =====")
        print("1. Add Student")
        print("2. Show All Students")
        print("3. Search Student")
        print("4. Delete Student")
        print("5. Exit")
        choice = input("Enter your choice: ")

    try:
        if choice == "1":
            add_student(students)
        elif choice == "2":
            show_students(students)
        elif choice == "3": ↓
            search_student(students)
        elif choice == "4":
            delete_student(students)
        elif choice == "5":
            save_students(students)
            print("💾 Data saved. Exiting program...")
            break
        else:
            print("✖ Invalid choice, try again.")
    except Exception as e:
        print("⚠ Error:", e)

    # -----
    # Run Program
    # -----
menu()

```

```
import csv

# -----
# File Handling Helpers
# -----
FILE_NAME = "students.csv"

def load_students():
    students = []
    try:
        with open(FILE_NAME, mode="r", newline="") as file:
            reader = csv.DictReader(file)
            for row in reader:
                students.append(row)
    except FileNotFoundError:
        pass # لو الملف مش موجود بيدأ البرنامج بليست فاضية
    return students
```



```
def save_students(students):
    with open(FILE_NAME, mode="w", newline="") as file:
        fieldnames = ["Name", "Age", "Grade"]
        writer = csv.DictWriter(file, fieldnames=fieldnames)
        writer.writeheader()
        writer.writerows(students)

# -----
# Core Functions
# -----
def add_student(students):
    name = input("Enter student name: ")
    age = input("Enter student age: ")
    grade = input("Enter student grade: ")
    students.append({"Name": name, "Age": age, "Grade": grade})
    print(f"✓ Student {name} added successfully!")
```



```
def show_students(students):
    if not students:
        print("X No students found.")
        return
    print("\n--- All Students ---")
    for s in students:
        print(f"{s['Name']} | Age: {s['Age']} | Grade: {s['Grade']}")

def search_student(students):
    name = input("Enter name to search: ")
    found = [s for s in students if s["Name"].lower() == name.lower()]
    if found:
        for s in found:
            print(f"✓ Found: {s['Name']} | Age: {s['Age']} | Grade: {s['Grade']}")
    else:
        print("X Student not found.")
```



```
def delete_student(students):
    name = input("Enter name to delete: ")
    for s in students:
        if s["Name"].lower() == name.lower():
            students.remove(s)
            print(f"✗ Student {name} deleted.")
            return
    print("X Student not found.")

# -----
```

