Introduction To Programming

Second Course

Lecture 3

Assist. Prof. Dr. Abdul Hadi Mohammed

Introduction to Files and Functions in Python

Functions in Python

Definition & Syntax

```
def function_name(parameters):
    # body of the function
    return value
```

Example: A simple function

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

print(greet("Ali"))
```

Working with Files in Python

Opening a File

```
file = open("example.txt", "r")  # modes: 'r', 'w', 'a', 'x'
```

File Modes in Python

Mode	Name	Description
`'r'`	**Read**	Opens file **for reading only**. File must exist.
`'w'`	**Write**	Opens file **for writing only**. **Overwrites** if file exists. Creates new file if not.
`'a'`	**Append**	Opens file **for appending**. Creates file if not exists. Adds to the end without deleting existing data.
`'x'`	**Create**	Creates new file. **Fails if file already exists** (throws `FileExistsError`).

Examples for Each Mode

`'r'` – Read mode

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

`'w'` – Write mode

```
# Overwrites file if exists
with open("data.txt", "w") as file:
    file.write("This will erase and write new content.")
```

`'a'` – Append mode

```
# Adds new data to the end of the file
with open("data.txt", "a") as file:
    file.write("\nAdding a new line at the end.")
```

`'x'` – Exclusive create mode

```
# Fails if the file exists
with open("new_file.txt", "x") as file:
    file.write("This file was created using 'x' mode.")
```

If `new_file.txt` already exists, you'll get:

```
FileExistsError: [Errno 17] File exists: 'new_file.txt'
```

Add 'b' for Binary or '+' for Read/Write

- `'rb'` → read binary (e.g. images, videos)
- `'wb'` → write binary
- `'r+'` → read and write
- `'a+'` → read and append

Reading from a File

```
content = file.read()
```

Writing to a File

```
file = open("output.txt", "w")
file.write("This is a line.\n")
file.close()
```

Using `with` Statement (Best Practice)

```
with open("output.txt", "a") as file:
    file.write("Another line.\n")
```

`'a'` vs `'w'` vs `'a+'`

Mode	Read	Write	File Created If Missing	Overwrites?	Appends At End	File Pointer Start
`'w'`	No	Yes	Yes	Yes (clears old data)	No	Start (0)
`'a'`	No	Yes	Yes	No (keeps data)	Yes	End of file
`'a+'`	Yes	Yes	Yes	No (keeps data)	Yes	End of file (but you can seek manually)

`'w'` – Write Only

- Creates a new file if it doesn't exist.
- **Erases everything** in the file if it already exists.
- You **cannot read** from the file.

```
with open("example.txt", "w") as f:
    f.write("New content")  # old content is deleted
```

'a' – Append Only

- Adds new content to the **end** of the file.
- You **cannot read** the file content.

```
with open("example.txt", "a") as f:
    f.write("\nMore data added")  # added at end
```

'a+' – Append and Read

- Same as `'a'` **but** you can **read the file too**.
- **File pointer starts at the end**, so reading needs `.seek(0)` if you want to read from the beginning.

```
with open("example.txt", "a+") as f:
    f.write("Appended line\n")
    f.seek(0)  # move to the beginning
    print(f.read())  # now you can read entire content
```

Summary in Code

```
with open("file.txt", "a+") as f:
    f.write("Hello\n")
    f.seek(0)
    print(f.read())
```

This works, but:

- `'a'` would fail at `f.read()`
- `'w'` would delete old content before writing

Q1: Write a function that reads a file and prints each line with line numbers.

Solution:

```
def print_file_with_line_numbers(filename):
    with open(filename, "r") as file:
        for i, line in enumerate(file, start=1):
            print(f"{i}: {line.strip()}")
# Test
print_file_with_line_numbers("sample.txt")
```

Q2: Write a function that counts the number of lines in a file and returns the result.

```
def count_lines(filename):
    with open(filename, "r") as file:
        return sum(1 for _ in file)
# Test
print("Number of lines:", count_lines("sample.txt"))
```

Q3: Write a function that takes a list of strings and writes each string into a file.

Solution:

```
def write_lines_to_file(lines, filename):
    with open(filename, "w") as file:
        for line in lines:
            file.write(line + "\n")
lines = ["First line", "Second line", "Third line"]
write_lines_to_file(lines, "output.txt")
```

Q4: Write a function that reads a file, converts all content to uppercase, and saves it to another file.

```
def convert_file_to_uppercase(input_file, output_file):
    with open(input_file, "r") as infile, open(output_file, "w") as
outfile:
        for line in infile:
            outfile.write(line.upper())
# Test
convert_file_to_uppercase("sample.txt", "uppercase.txt")
```

Q5: Write a program that asks the user to enter 3 names and stores them in a file using a function.

```
def save_names(filename):
    with open(filename, "w") as file:
        for _ in range(3):
            name = input("Enter a name: ")
            file.write(name + "\n")
# Call the function
save_names("names.txt")
```

Strings

String Methods:

```
text = "  Hello Python  "
print(text.strip())       # Remove spaces
print(text.upper())       # Convert to UPPERCASE
print(text.lower())       # Convert to lowercase
print(text.replace("Python", "World"))  # Replace word
```

Question 1: Count vowels in a string using a function

```
def count_vowels(s):
    vowels = "aeiouAEIOU"
    return sum(1 for char in s if char in vowels)

print(count_vowels("Welcome to Python"))
```

Lists

```
fruits = ["apple", "banana", "cherry"]
fruits.append("mango")
fruits.remove("banana")
print(fruits[0])  # Access first item
```

Question : Write a function to read lines from a file into a list

```
def file_to_list(filename):
    with open(filename, "r") as f:
        return [line.strip() for line in f]

print(file_to_list("names.txt"))
```

Dictionaries

```
student = {"name": "Ali", "age": 20, "grade": "A"}
print(student["name"])
student["age"] = 21
student["gender"] = "male"
```

Question : Save student data to file and load it

```
def save_students(students, filename):
    with open(filename, "w") as f:
        for s in students:
            line = f"{s['name']},{s['age']},{s['grade']}\n"
            f.write(line)

def load_students(filename):
    with open(filename, "r") as f:
        return [dict(zip(["name", "age", "grade"],
line.strip().split(","))) for line in f]

# Example usage
students = [{"name": "Sara", "age": "19", "grade": "A"}, {"name":
"Omar", "age": "20", "grade": "B"}]
save_students(students, "students.txt")
print(load_students("students.txt"))
```

Question: Function to reverse a string

```
def reverse_string(s):
    return s[::-1]

print(reverse_string("python"))
```

Question: Function that returns the longest word in a list

```
def longest_word(words):
    return max(words, key=len)

print(longest_word(["apple", "banana", "kiwi", "strawberry"]))
```

Question: Dictionary of word counts from a text file

```
def word_count(filename):
    counts = {}
    with open(filename, "r") as f:
        for line in f:
            for word in line.strip().split():
                word = word.lower()
                counts[word] = counts.get(word, 0) + 1
    return counts

print(word_count("sample.txt"))
```

Assignment :

1. Create a dictionary from a text file where each key is a unique word and the value is the number of times it appears.
2. Write a function that reads two files, combines their content, and writes it to a third file.
3. Write a function that asks the user to enter student names and grades, saves them in a file, and another function that reads the file and prints the student names with their grades.