Unit:2 ; Class:10

Files, Modules, and Packages

Instructor: Musfique Ahmed

Introduction to Files

- Files are used to store data permanently.
- Common operations include reading from and writing to files.
- File types: Text files (`.txt`) and Binary files.

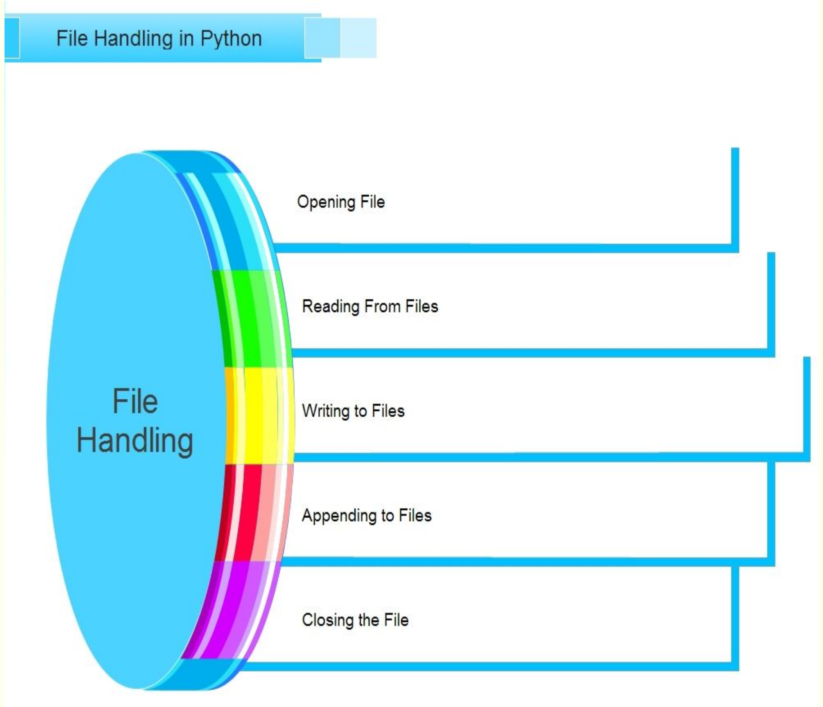
File Handling Basics

1. Open a file using `open()`:

```
file = open('filename.txt', 'mode')
```

2. Modes of operation:
   - `'r'`: Read (default)
   - `'w'`: Write
   - `'a'`: Append
   - `'b'`: Binary
3. Always close the file:

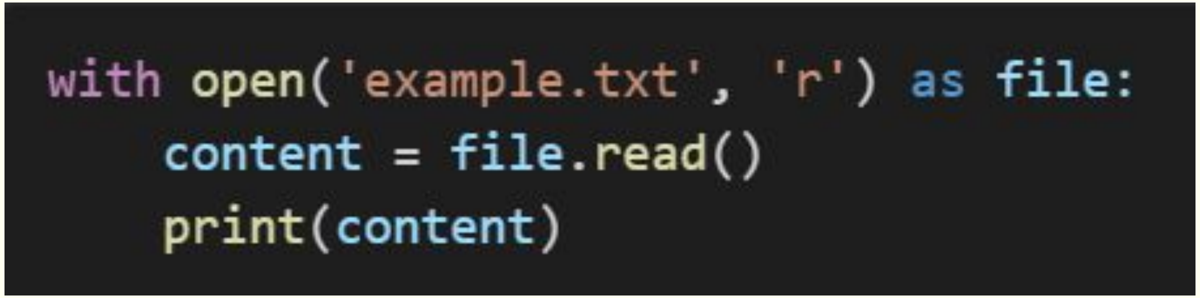
```
file.close()
```



Practice Problem: Open a text file named `example.txt` and print its contents.

Reading from Files

- Use `read()` to read the entire file.
- Use `readline()` to read one line at a time.
- Use `readlines()` to get a list of all lines.



```python
with open('example.txt', 'r') as file:
    content = file.read()
    print(content)
```

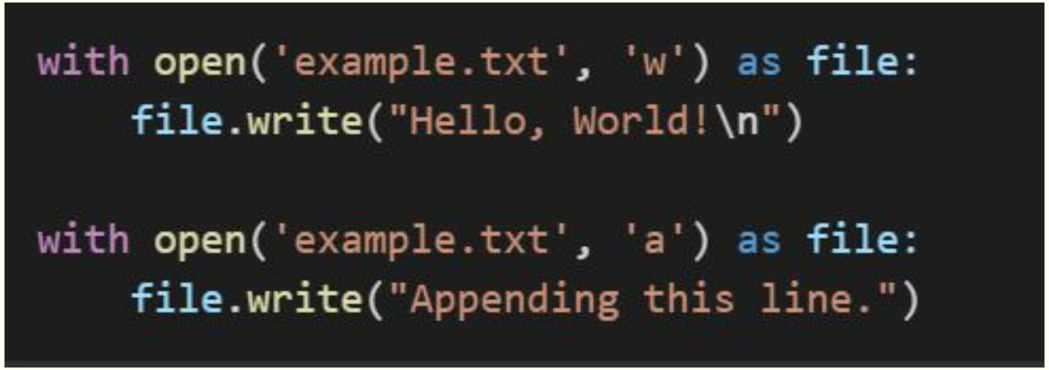
Practice Problem: Write code to read a file and count the number of lines in it.

Steps

1. Create a .txt file. (ex- my_info.txt)
2. Note: always save the .txt file [ctrl + s]
3. Create a python file
4. File = open(“file_path”, “r”)
5. Note: always make all \ into / (\ ->/)
6. Content = file.read()
7. print(content)
8. Impt: close the file [file.close()]

Writing and Appending to Files

- Use `'w'` mode to overwrite a file.
- Use `'a'` mode to append to a file.



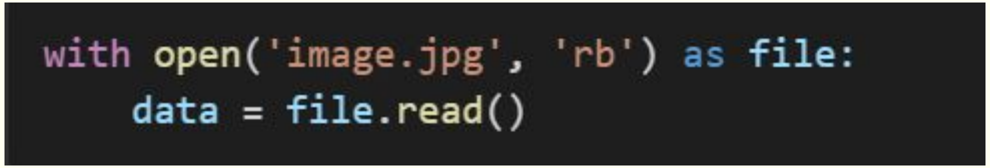
```python
with open('example.txt', 'w') as file:
    file.write("Hello, World!\n")

with open('example.txt', 'a') as file:
    file.write("Appending this line.")
```

Practice Problem: Write a Python program to write a list of strings to a file, each on a new line.

Working with Binary Files

- Used to handle non-text files (e.g., images, videos).
- Open in binary mode (`'rb'`, `'wb'`, `'ab'`).

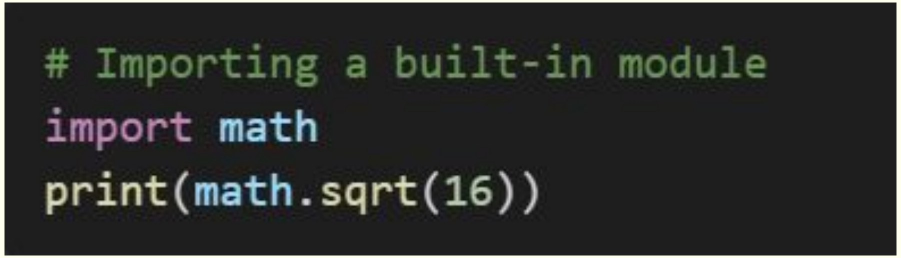


```
with open('image.jpg', 'rb') as file:
    data = file.read()
```

Practice Problem: Write a program to copy the contents of one binary file to another.

Introduction to Modules

- A module is a file containing Python code (functions, variables, classes) that can be reused.
- Built-in modules: Provided by Python (e.g., `math`, `os`, `random`).
- User-defined modules: Created by programmers.



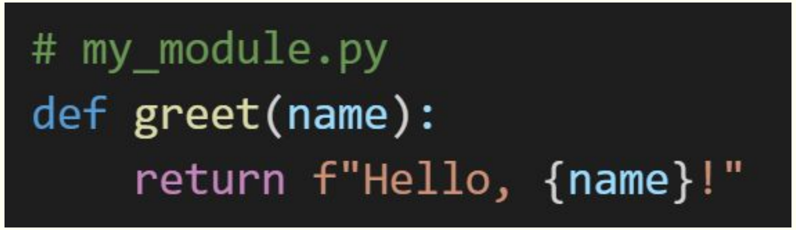
```python
# Importing a built-in module
import math
print(math.sqrt(16))
```

Practice Problem: Import the `random` module and generate a random number between 1 and 100.

Creating and Importing Modules

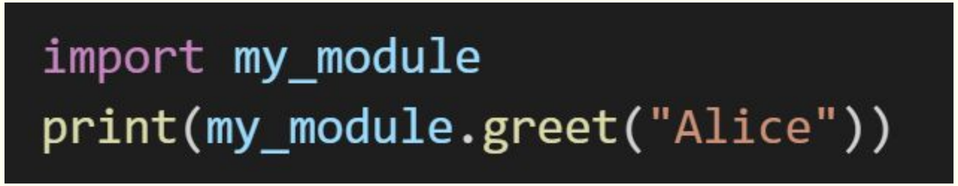
User-Defined Modules:

1. Create a Python file (e.g., `my_module.py`) with the following code:



```python
# my_module.py
def greet(name):
    return f"Hello, {name}!"
```

2. Import and use it:

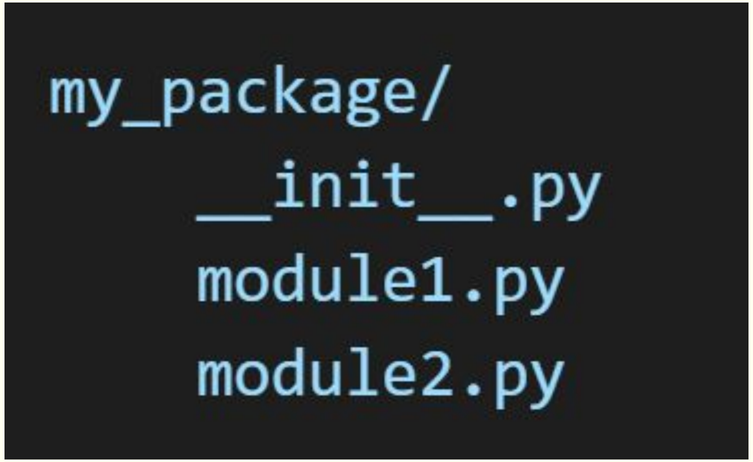


```python
import my_module
print(my_module.greet("Alice"))
```

Practice Problem: Create a module with a function to calculate the factorial of a number. Import and test it.

What is a Package?

- A package is a collection of modules organized in directories.
- Must include an `__init__.py` file (can be empty).

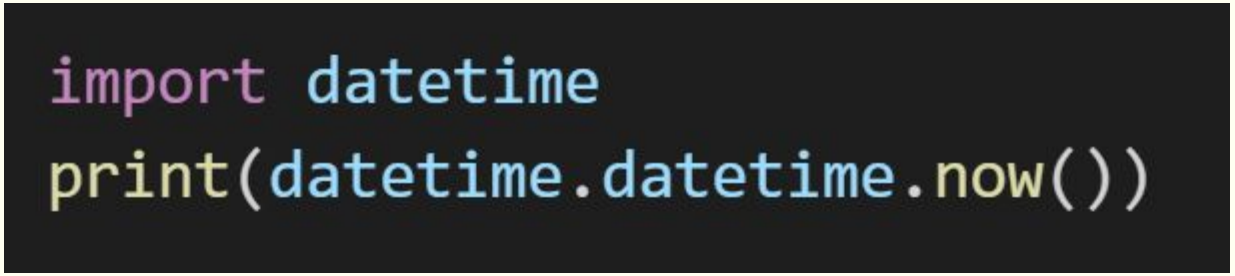


```
my_package/
    __init__.py
    module1.py
    module2.py
```

Practice Problem: Create a package with two modules: one for math operations and another for string operations.

Commonly Used Built-in Modules

1. `math`: Mathematical functions
2. `os`: Operating system interfaces
3. `sys`: System-specific parameters and functions
4. `random`: Generate random numbers
5. `datetime`: Work with dates and times



```python
import datetime
print(datetime.datetime.now())
```

Practice Problem: Use the `datetime` module to print yesterday's date.

Different Types of Files

1. **Text Files (``):** Store plain text data. Easy to read and write.
   - Example: Logs, notes.
2. **CSV Files (``):** Store tabular data in a comma-separated format.
   - Example: Spreadsheets, databases.
3. **Excel Files (``):** Used for structured data with formatting and formulas.
   - Example: Reports, analytics.
4. **JSON Files (``):** Store data in key-value pairs for web and API interactions.
   - Example: Config files, API responses.
5. **Binary Files:** Store non-text data such as images, audio, or video.
   - Example: Multimedia files.

Practice Problem: Create a program to read a CSV file and print its contents line by line.

Summary

- File handling basics: Reading, writing, and appending.
- Modules: Built-in and user-defined.
- Packages: Organizing multiple modules.
- Practice using built-in modules like `math` and `datetime`.

Practice Problem: Reflect on the lesson by writing a program that combines file handling and modules (e.g., read a file and perform calculations using `math`).

Topics We Have Covered:

1. **Lists**
   - Indexing, adding/removing elements, modifying elements, looping, nested structures.
2. **Conditionals**
   - `if`, `elif`, `else` statements.
3. **Loops**
   - `for` loops, `while` loops, using `break` and `continue`.
4. **Dictionary**
   - Creating, updating, deleting keys, nested dictionaries, and looping through dictionaries.
5. **Sets**
   - Operations like union, intersection, and difference.
6. **Functions**
   - Creating functions, keyword and positional arguments, built-in functions.
7. **Files and Modules**
   - File operations (`open`, read/write), creating and importing modules, packages, and using built-in modules.
8. **Different File Types**
   - `.txt`, `.csv`, `.xlsx`, and binary files.
9. **Tuples**
   - Immutable lists, basic operations, and use cases.
10. **Basic Data Types and Casting**
    - Integers, floats, booleans, and casting between types (e.g., `int()`, `str()`).

Possible Missing Basics:

1. **String Manipulation**
   - String methods (`split`, `strip`, `replace`, etc.), slicing, and formatting (`f-strings` or `.format()`).
2. **List Comprehensions**
   - Simplified syntax for creating lists.
3. **Python Debugging Tools**
   - `print()` debugging and using the `pdb` module for step-by-step debugging (if relevant).
4. **Logical and Membership Operators**
   - `and`, `or`, `not`, `in`, and `not in`.

---

5. **Intro to Object-Oriented Programming (OOP)**
   - Classes, objects, and simple methods (optional for beginners).
6. **Error and Exception Handling**
   - Using `try`, `except`, and `finally` blocks.

Thank You

Do the Quiz Please, you have 10 minutes to do that!