## Chapter 2.



# Python Programming

## UNIT 3. Python III

- 3.1. Control Structures.
- 3.2. Python Functions.
- 3.3. Python Input and Output.

## Unit 4. Python IV

- 4.1. Classes and Objects.
- 4.2. Exception Handling.

## Unit 5. Python V

- 5.1. Algorithms.
- 5.2. Data Structures.
- 5.3. Working with Files.
- 5.4. Working with Excel, Word, PDF Documents.



# Working with Files (1/8)

#### Using the os library:

Different operating systems use different path separators.

import os os.path.join(str1, str2, str3,)	# A complete path by joining strings.
print(os.path.sep)	# Path separator.



# Working with Files (2/8)

- Using the os library:
  - Current working directory and directory change.

```
print(os.getcwd()) # Current working directory.
os.chdir(<str_path>) # Change the working directory to <str_path>.
```



# Working with Files (3/8)

- Using the os library:
  - Absolute path.

```
# Make an absolute path based on the current working directory.

str_path_abs = os.path.abspath("my_file.txt")

print(str_path_abs)
```

```
# Bring the file name from an absolute path.
print(os.path.basename(str_path_abs))
# Bring the directory structure from an absolute path.
print(os.path.dirname(str_path_abs))
```

```
print(os.path.isabs(str_path))
```

# Check whether str\_path is an absolute path.



# Working with Files (4/8)

- Using the os library:
  - How to check whether the path points to a folder or a file.

```
# Check whether the path points to a folder.
print(os.path.isdir(str_path))
print(os.path.isdir(str_path_abs))
```

```
# Check whether the path points to a file.
print(os.path.isfile(str_path))
print(os.path.isfile(str_path_abs))
```



# Working with Files (5/8)

- Using the os library:
  - List the content of a folder.

```
list_dir = os.listdir()  # Files and subfolders of the current working directory.
list_dir.sort()  # Sort the listing.
```

```
# Show only those files of which names start with 'c' or 'C'.
for x in list_dir:
   if x.lower()[0] == 'c':  # Lower case first character matching with 'c'.
        print(x)
```



# Working with Files (6/8)

#### Using the pickle library:

Store an object in an external file and then restore it later.

```
import pickle
x = [1,2,3, {'Name':'James', 'Age':30, 'Height':180}]  # A composite object.
pickle.dump(x, open('my_pickle.pkl','wb'))  # Store object x in an external file.
del x  # Delete the object x.
new_x = pickle.load(open('my_pickle.pkl','rb'))  # Bring back the stored object.
print(new_x)
```



# Working with Files (7/8)

- Using the shelve library:
  - Store data in an external file as a dictionary and then restore it later.

```
import shelve
# Store.
x = shelve.open('MyDict')  # 3 binary files are created: .bak, .dat, .dir
x['Name'] = 'James'  # A key:value pair.
x['Age'] = 30
x['Height'] = 180
x.close()  # Close.
```



# Working with Files (8/8)

- Using the shelve library:
  - Store data in an external file as a dictionary and then restore it later.

```
# Read in and restore.
x = shelve.open('MyDict')
print(list(x.keys()))
print(list(x.values()))
print(list(x.items()))
x.close()
```



## Coding Exercise #0111



## Chapter 2.



# Python Programming

## UNIT 3. Python III

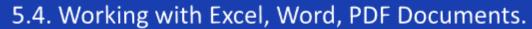
- 3.1. Control Structures.
- 3.2. Python Functions.
- 3.3. Python Input and Output.

## Unit 4. Python IV

- 4.1. Classes and Objects.
- 4.2. Exception Handling.

## Unit 5. Python V

- 5.1. Algorithms.
- 5.2. Data Structures.
- 5.3. Working with Files.
- 5.4. Working with Excel, Word, PDF Documents.





# Working with Excel Documents (1/3)

### Using the openpyxl library:

Workbook > Sheet > Cell.

```
import openpyxl
wb = openpyxl.load_workbook('my_excel.xlsx')  # Open a workbook.
wb.sheetnames  # List of the sheet names.

sh = wb['Sheet1']  # A sheet object pointing to 'Sheet1'.
cl = sh['A1']  # A cell object pointing to 'A1'.
print(cl.value)
print(sh['A1'].value)  # Value of the cell 'A1'.
print(sh.cell(1,1).value)  # Value of the cell 'A1' by the coordinates.
```

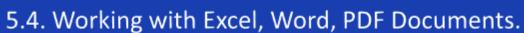




# Working with Excel Documents (2/3)

- Using the openpyxl library:
  - Workbook > Sheet > Cell.

# Display values from several cells. for i in range(1,11): print(sh.cell(i,1).value)





# Working with Excel Documents (3/3)

### Using the openpyxl library:

Workbook > Sheet > Cell.

```
# Create a new workbook.

my_wb = openpyxl.Workbook()  # This workbook only exists in the memory.

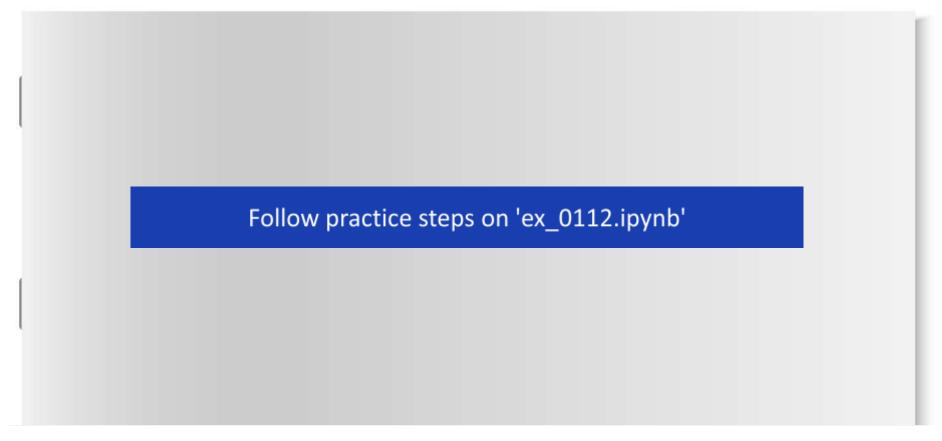
print(my_wb.sheetnames)  # Only 'Sheet' exists in the new workbook.
```

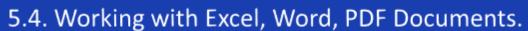
```
my_sh = my_wb['Sheet']
my_sh['A1'].value = 999  # Enter a new value.
my_sh['A2'] = 666  # This is also OK.
my_sh.title = 'MySheet1'  # Change the sheet name.
my_sh2 = my_wb.create_sheet(index = 0, title = 'MySheet2')
my_wb.save('my_new_excel.xlsx')  # The workbook is saved in an external file.
```





## Coding Exercise #0112







# Working with Word Documents (1/3)

### Using the docx library:

Document > Paragraph > Run.

```
import docx
my_doc = docx.Document('What is Design Thinking.docx') # Open a word document.
```

```
n = len(my_doc.paragraphs) # Number of the paragraphs.

print(n)

print(my_doc.paragraphs[0].text) # Text of the paragraph 0.

print(my_doc.paragraphs[11].text) # Text of the paragraph 11.

print(my_doc.paragraphs[33].text) # Text of the paragraph 33.
```





# Working with Word Documents (2/3)

### Using the docx library:

Document > Paragraph > Run.

# A new run starts when there is a style change.  m = len(my_doc.paragraphs[33].runs)  print(m)	# Number of runs in a paragraph.
print(my_doc.paragraphs[33].runs[10].text)	# Text content from a specific run.



## 5.4. Working with Excel, Word, PDF Documents.

## Working with Word Documents (3/3)

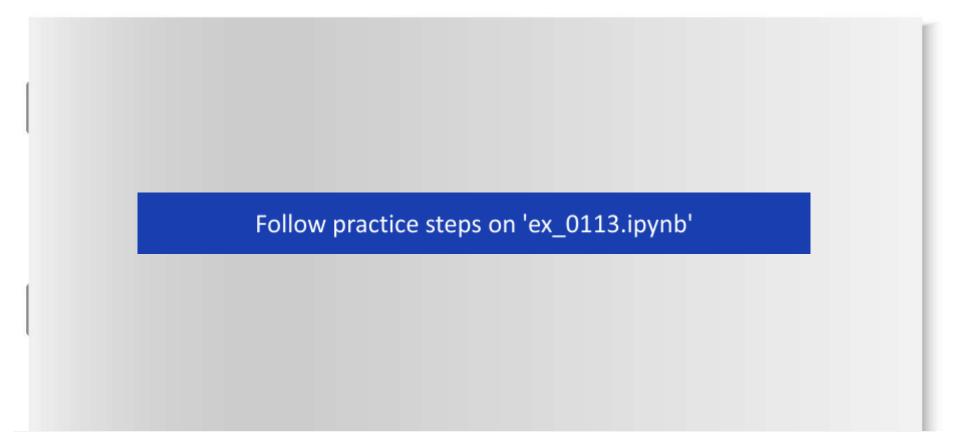
- Using the docx library:
  - Document > Paragraph > Run.

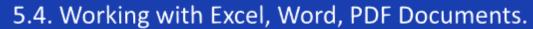
```
# Create a new Word document.
my_new_doc = docx.Document()
# Add new paragraphs.
my_new_doc.add_paragraph("My first paragraph!")
my_new_doc.add_paragraph("My second paragraph!")
my_new_doc.add_paragraph("My third paragraph!")
# Save the document to an external file.
my_new_doc.save("my_new_doc.docx")
```





## Coding Exercise #0113







## Working with PDF Documents

## Using the PyPDF2 library:

```
import PyPDF2
# Open in binary read mode.
my_doc = open('my_document.pdf', 'rb')
# Create reader object.
my_reader = PyPDF2.PdfFileReader(my_doc)
n = my_reader.numPages # Number of pages.
```

```
# Read a page from the PDF document.

my_page = my_reader1.getPage(17)  # Get a specific page.

print(my_page.extractText())  # Extract text (may or may not work).
```





## Coding Exercise #0114

