Python: Input & Output

1405 Instructor: Ruiqing (Sam) Cao

Print On Screen

- You can print *any data object* on the screen
  - String `print('Alice')` list `print([1,2,3])` dict `print({1:'a'})`
  - Numeric type `print(3.14)` `print(10)` Boolean `print(True)`

- You can *print multiple objects* separated by comma (,)
  - `print('The result is:', 3.14)`

Print Formatting

This applies to string formatting in general: you can insert value or variable (and their appropriate formats) into a string

Example: print the base of natural log up to 2 decimal places

Two approaches (and `fstring` increasingly more common)

- `str.format()`: `print("e equals {:.2f}".format(2.71828))`
- `fstring`: `print(f"e equals {2.71828:.2f}")`

In both approaches the double quotes `""` can be replaced by single quotes `''` without any difference

Input Data from Keyboard

```
data = input("Enter a number here:")
print(f"The number entered is {data}")
```

- `input()` ***displays the prompt string*** `"Enter a number here:"` and waits for the user to ***type an input and press Enter***

- The input is captured as a string and stored in the variable `data`

Open & Close a File

```
f = open(filename, mode)
```

`filename`: the path and name of the file to open

`mode`: the mode for opening the file (`'r'` read, `'w'` write, `'a'` append)

```
f.close()
```

- Close an existing file `f` that is already open
- The operation is ignored if the file `f` is already closed (but no error)

Open & Close a File

- You should make sure a file is closed after you finish, or use the `with` statement to ensure files are automatically closed

```
with open (filename, mode) as f:
    ...
```

- If you do not use the with statement, you must call `close()` to close the file manually

File Input: Read & Write a CSV File

- To read a `f`'s content, it must be already open in *read* (`'r'`) mode
- To write content to `f`, it must be already open in *write* (`'w'`) mode
- The Python module `csv` is widely used to handle CSV files that are relatively *simple* (no complex types causing problems for parsing)

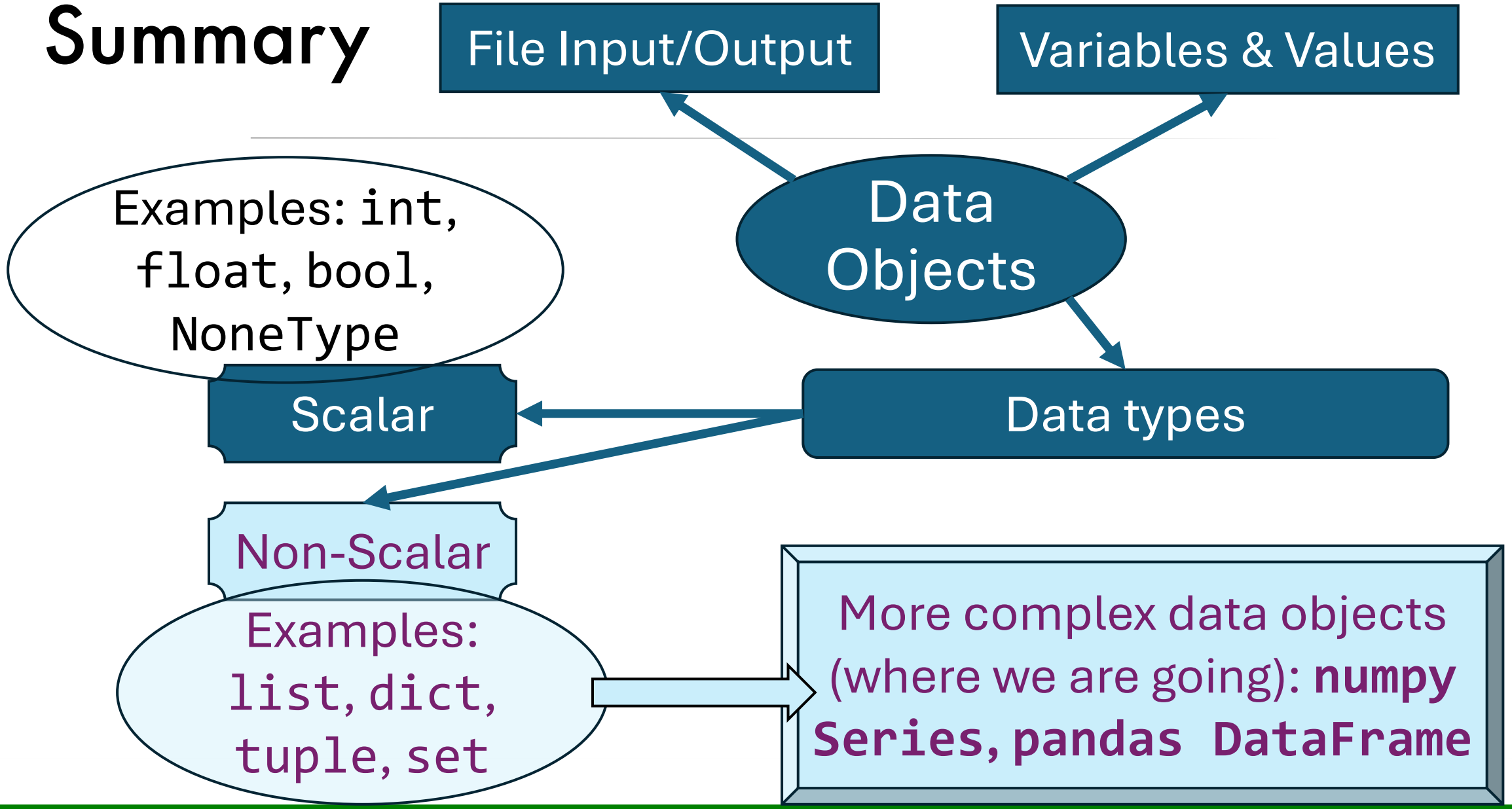
Reading data from a CSV file:

```
import csv
with open("dummy.csv", "r") as f:
    f_csv = csv.reader(f)
    for row in f_csv:
        print(row)
```

Writing data to a CSV file:

```
import csv
data = [['id','firm'],[10,'KKR'],[11,'HP']]
with open("dummy.csv", "w") as f:
    f_csv = csv.writer(f)
    for row in data:
        f_csv.writerow(row)
```

Summary


File Input/Output
Variables & Values
Data Objects
Examples: int, float, bool, NoneType
Scalar
Data types
Non-Scalar
Examples: list, dict, tuple, set
More complex data objects (where we are going): numpy Series, pandas DataFrame