Reading/Writing Data

Shankar Kulumani

Flight Dynamics & Control Lab

THE GEORGE WASHINGTON UNIVERSITY

WASHINGTON, DC

Why import/export?

- Most data will not be “hard-coded” into your programs
  - Data from observations to process
  - Data from models to export and analyze
- Allow for multiuse programs by reading in data from the user - interaction

Getting data from the user

- Prompt the user to input data

```
a = input("What is the first number? ")
b = input("What is the second number? ")

a = int(a)
b = int(b)
```

- Input data is converted to a string - you can convert it to a different data type as desired

Importing text from files

- First must “open” the file for reading

  ```
  file = open("filename.txt", "r")
  ```

- There are many ways to read a text file in Python
  - Reading data by the line - Useful for sequential data (tracking station data)

    ```
    print(file.readline())
    ```

  - Read a big block of data - Processing data for analysis (missle intercept data)

    ```
    print(file.read())
    ```

- After reading we must “close” the file

  ```
  file.close()
  ```

Opening a file

- Before reading/writing to a file it must be opened
- This will create a file object which we can then operate on

  ```
  file_object = open("filename", "mode")
  ```

- Several ways to open a file
  - `'r'` - Read mode which the file is only read
  - `'w'` - Wirte mode which is used to edit and write new data - will erase whatever is in `"filename"`
  - `'a'` - Append mode which is used to add data to the end of the file
  - `'r+'` - Read and write mode - both combined into one

Reading by line

- Can use a loop to read a file line by line and print the output

```
file = open("testfile.txt", 'r')
for line in file:
    print(line)
file.close()
```

- Convienent to use the `with` statement

```
with open("testfile.txt", 'r') as file:
    for line in file:
        print(line)
```

- `with` will automatically close the file for us

Exporting data

- After performing calculations we'd like to have a way to save the results to a file for later use
- Two different ways to save file - among many, many others
  - Write to text files
  - Save to a `numpy` file

Writing text to a file

- Can write strings to a text file

```
with open("textfile.txt", 'a') as file:
    file.write("This is a string")
    file.write("Here is another string")
```

Writing to a data file

- We can save data “as-is” for later use
- Preserve arrays/lists etc

```
import numpy as np
a = np.arange(10)
b = np.arange(100)
np.save("outfile", (a, b))
```

Exercise: Import and export data

- There is a file caled `vector.txt`
  - 6 numbers which represent 3 position and 3 velocity components all in a single line
- Write a function `getposvel` that will:
  - Import the first three numbers into a position array
  - Import last three numbers into a velocity array
  - Inputs: filename to open, Outputs: `pos, vel` the output arrays

Exercise: Testing

- Your code should be tested. It should at a minimum pass the following unit tests

```
def test_getposvec_array_sizes():
    pos, vel = getposvec('vector.txt')
    np.testing.assert_allclose(pos.shape, (3,))
    np.testing.assert_allclose(vel.shape, (3,))
```

- Determine what other tests you need to verify your function

Exercise: Output

- Write a function `writevec` that will:
  - Create a `*.txt` file with the user's desired name
  - Label the components of a vector with the user's choice
  - Write the components of a vector, with label to a the text file (x, y, z or i, j, k, etc)
  - Inputs: filename, vector, array of labels
  - Outputs: Success or failure flag