TUS

Topic 5
Files and Serialisation

Dr Diarmuid Ó Briain

Module Objectives

On completion of this module the learner will/should be able to:

- Testing Files and Directories
- Writing to and Deleting files
- Reading from files
- Importing Structured Data
- Interact with Spreadsheets
- Apply Serialisation Protocols; YAML, JSON

Testing Files and Directories

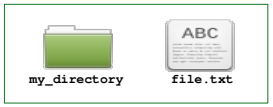
TUS

Testing if files or directories exist


```
~$ cat exists.py
#!/usr/bin/env python3

# Do this in the shell first
#  ~$ mkdir my_directory
#  ~$ echo "FILE" > my_directory/file.txt

from os import path

import sys

my_dir = "my_directory"
my_file = "my_directory/file.txt"
other_file = "my_directory/file2.txt"

print(f"'{my_dir}' exists: {path.exists(my_dir)}")
print(f"'{my_file}' exists: {path.exists(my_file)}")
print(f"'{other_file}' exists: {path.exists(other_file)}")
print(f"'{my_dir}' is a directory: {path.isdir(my_dir)}")
print(f"'{my_file}' is a directory: {path.isdir(my_file)}")
print(f"'{my_dir}' is a file: {path.isfile(my_dir)}")
print(f"'{my_file}' is a file: {path.isfile(my_file)}")

# // End //
sys.exit(0)

```

Testing if files or directories exist

```
~$ mkdir my_directory

~$ echo "FILE" > my_directory/file.txt

~$ ./exists.py
'my_directory' exists: True
'my_directory/file.txt' exists: True
'my_directory/file2.txt' exists: False
'my_directory' is a directory: True
'my_directory/file.txt' is a directory: False
'my_directory' is a file: False
'my_directory/file.txt' is a file: True
```



Writing to files

```
~$ cat write2file.sh
 1  #! /usr/bin/env python3
 2
 3  import sys
 4
 5  # // File for writing //
 6  some_text = "Some more text"
 7
 8  # // Writing to a file //
 9  fh = open("file.txt", mode="w", encoding="utf-8")
10  fh.write("This is a test file\nthat I want to write to")
11  fh.write("\n")
12  fh.write(some_text)
13  fh.write("\nfinished.")
14  fh.write("\n")
15  fh.close()
16
17  # // End //
18  sys.exit(0)
19
```

```
~$ ./write2file.py

~$ cat file.txt
This is a test file
that I want to write to
Some more text
finished.
```

Modes

r	Read	r+	Read & Write
w	Write	w+	Read & Write
a	Append	a+	Read & Write

Deleting files

```
~$ ls my_directory/file.txt
my_directory/file.txt


>>> import os

>>> if (os.path.isfile("my_directory/file.txt")):
...     print("File file.txt exists")
...     os.remove("my_directory/file.txt")
...     print("Deleted file.txt")
...
File file.txt exists
Deleted file.txt

>>> quit()


~$ ls my_directory/file.txt
ls: cannot access 'my_directory/file.txt': No such file or directory
```

Reading from files

TUS

Reading files as a string (`read()`)

```
~$ cat readafile_as_string.py
 1  #! /usr/bin/env python3
 2
 3  import sys
 4
 5  # // File for reading //
 6  file = "unstructured_data.txt"
 7
 8  print()
 9
10  # // Read entire file //
11  print("** Reading entire file, this is the output **\n")
12  fh = open(file, mode="r")
13  print((type(fh), fh))
14  entire_file = fh.read()
15  print(entire_file)
16  print()
17
18  # // Closing filehandle //
19  fh.close()
20
```


Reading files as a string

```
~$ ./readafile_as_string.py

** Reading entire file, this is the output **

(<class '_io.TextIOWrapper'>, <_io.TextIOWrapper name='unstructured_data.txt' mode='r' encoding='UTF-8'>)
Port      Status    Vlan  Duplex      Speed  Type
Gi0/1/0   connected   1   auto        auto   10/100/1000BaseTX
Gi0/1/1   connected   1   half-duplex 100    10/100/1000BaseTX
Gi0/1/2   notconnect  1   full-duplex 1000   10/100/1000BaseTX
Gi0/1/3   notconnect  1   auto        auto   10/100/1000BaseTX
```


Reading files as a string (`read()`, `splitlines()`)

```
21  # // Read entire file (alternative format) //
22  print()
23  print("** Reading entire file, this is the output **\n")
24  with open(file) as fh2:
25      print((type(fh2), fh2))
26      entire_file2 = fh2.read()
27  print(entire_file2)
28
29  try:
30      entire_file3 = fh2.read()
31  except:
32      print("\nfilehandle 'fh2' was closed with the indent closure\n")
33
34  print("** splitlines() for some structure **\n")
35  structured_list = entire_file.splitlines()
36  print(structured_list)
37
38  # // End //
49  sys.exit(0)
40
```

Reading files as a string (`splitlines()`)

```
~$ ./readafile_as_string.py

** Reading entire file, this is the output **

(<class '_io.TextIOWrapper'>, <_io.TextIOWrapper name='unstructured_data.txt' mode='r' encoding='UTF-8'>)
Port      Status    Vlan  Duplex      Speed  Type
Gi0/1/0   connected   1   auto        auto   10/100/1000BaseTX
Gi0/1/1   connected   1   half-duplex 100    10/100/1000BaseTX
Gi0/1/2   notconnect  1   full-duplex 1000   10/100/1000BaseTX
Gi0/1/3   notconnect  1   auto        auto   10/100/1000BaseTX

** Reading entire file, this is the output **

(<class '_io.TextIOWrapper'>, <_io.TextIOWrapper name='unstructured_data.txt' mode='r' encoding='UTF-8'>)
Port      Status    Vlan  Duplex      Speed  Type
Gi0/1/0   connected   1   auto        auto   10/100/1000BaseTX
Gi0/1/1   connected   1   half-duplex 100    10/100/1000BaseTX
Gi0/1/2   notconnect  1   full-duplex 1000   10/100/1000BaseTX
Gi0/1/3   notconnect  1   auto        auto   10/100/1000BaseTX

filehandle 'fh2' was closed with the indent closure

** splitlines() for some structure **

['Port      Status    Vlan  Duplex      Speed  Type ', 'Gi0/1/0   connected   1   auto        auto
10/100/1000BaseTX', 'Gi0/1/1   connected   1   half-duplex 100    10/100/1000BaseTX', 'Gi0/1/2
notconnect  1   full-duplex 1000   10/100/1000BaseTX', 'Gi0/1/3   notconnect  1   auto        auto
10/100/1000BaseTX']
```


Reading files line by line (`readlines()`)

```
~$ cat readafile_line_by_line.py
 1  #! /usr/bin/env python3
 2
 3  import sys
 4  from pprint import pprint
 5
 6  # // File for reading //
 7  file = "unstructured_data.txt"
 8  structured_list = list()
 9
10  print()
11
12  # // Read unstructured data file line by line to list //
13  with open(file) as fh2:
14      entire_file_as_list = fh2.readlines()
15
16  # // Output less-structured data //
17  print(type(entire_file_as_list), entire_file_as_list)
18
19  # // Process data in list //
20  for line in entire_file_as_list:
21      structured_list.append(line.strip().split())
22
23  # // Output structured data //
24  print()
25  pprint(structured_list)
26
27  # // End //
28  sys.exit(0)
29
```


Reading files line by line (`readlines()`)

```
~$ ./readafile_line_by_line.py

<class 'list'> ['Port      Status    Vlan  Duplex      Speed  Type \n', 'Gi0/1/0
connected   1   auto        auto   10/100/1000BaseTX\n', 'Gi0/1/1   connected   1
half-duplex 100    10/100/1000BaseTX\n', 'Gi0/1/2   notconnect  1   full-duplex 1000
10/100/1000BaseTX\n', 'Gi0/1/3   notconnect  1   auto        auto   10/100/1000BaseTX']

[['Port', 'Status', 'Vlan', 'Duplex', 'Speed', 'Type'],
 ['Gi0/1/0', 'connected', '1', 'auto', 'auto', '10/100/1000BaseTX'],
 ['Gi0/1/1', 'connected', '1', 'half-duplex', '100', '10/100/1000BaseTX'],
 ['Gi0/1/2', 'notconnect', '1', 'full-duplex', '1000', '10/100/1000BaseTX'],
 ['Gi0/1/3', 'notconnect', '1', 'auto', 'auto', '10/100/1000BaseTX']]
```


Importing structured data

```
~$ cat structured_data_mod.py
str_intf = """Port      Name   Status       Vlan   Duplex Speed Type
Gi0/1/0          connect    1     auto   auto  10/100/1000BaseTX
Gi0/1/1          connect    1     half-duplex   100   10/100/1000BaseTX
Gi0/1/2          notconnect   1     full-duplex   1000   10/100/1000BaseTX
Gi0/1/3          notconnect   1     auto   auto  10/100/1000BaseTX"""


list_intf = ["Gi0/1/0", "Gi0/1/1", "Gi0/1/2", "Gi0/1/3"]


dict_intf = {"Gi0/1/0": "connect", "Gi0/1/1": "connect",
             "Gi0/1/2": "notconnect", "Gi0/1/3": "notconnect"}
```


Importing structured data

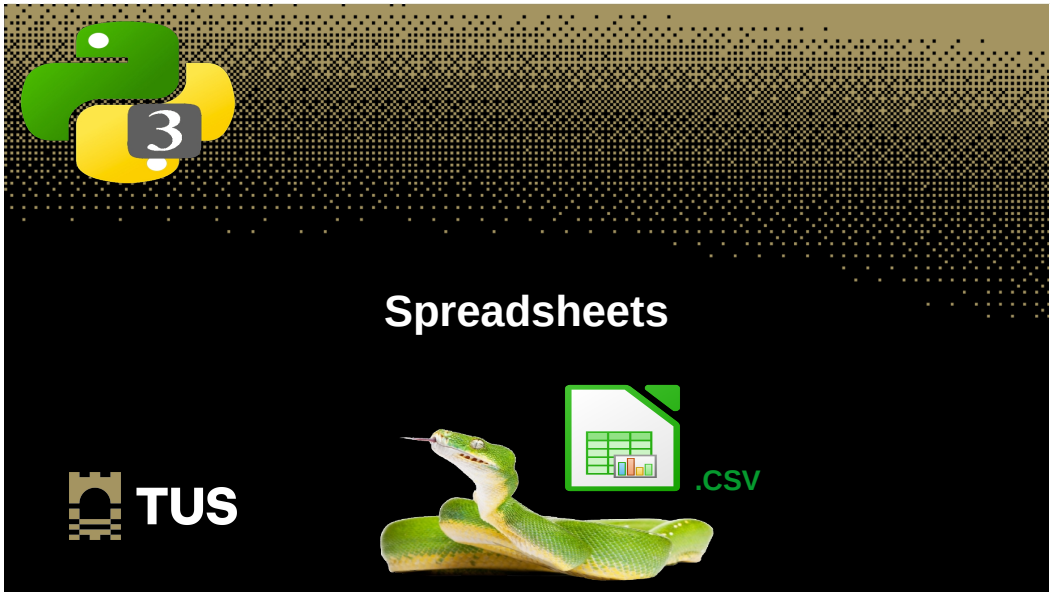
```
~$ python3
>>> import structured_data_mod as sdm


>>> sdm.str_intf
'Port      Name   Status       Vlan   Duplex Speed Type \nGi0/1/0          connect    1
auto   auto  10/100/1000BaseTX\nGi0/1/1          connect    1     half-duplex   100
10/100/1000BaseTX\nGi0/1/2          notconnect   1     full-duplex   1000
10/100/1000BaseTX\nGi0/1/3          notconnect   1     auto   auto  10/100/1000BaseTX'


>>> sdm.list_intf
['Gi0/1/0', 'Gi0/1/1', 'Gi0/1/2', 'Gi0/1/3']


>>> sdm.dict_intf
{'Gi0/1/0': 'connect', 'Gi0/1/1': 'connect', 'Gi0/1/2': 'notconnect', 'Gi0/1/3':
'notconnect'}
```


Importing complex structured data

```
list_of_lists = [['PORT_NAME', 'STATUS', 'VLAN', 'DUPLEX', 'SPEED', 'TYPE'],
['Gi0/1/0', 'connect', '1', 'auto', 'auto', '10/100/1000BaseTX'],
['Gi0/1/1', 'connect', '1', 'half-duplex', '100', '10/100/1000BaseTX'],
['Gi0/1/2', 'notconnect', '1', 'full-duplex', '1000', '10/100/1000BaseTX'],
['Gi0/1/3', 'notconnect', '1', 'auto', 'auto', '10/100/1000BaseTX']]


dict_of_lists = {'Gi0/1/0': ['connect', '1', 'auto', 'auto', '10/100/1000BaseTX'],
'Gi0/1/1': ['connect', '1', 'half-duplex', '100', '10/100/1000BaseTX'],
'Gi0/1/2': ['notconnect', '1', 'full-duplex', '1000', '10/100/1000BaseTX'],
'Gi0/1/3': ['notconnect', '1', 'auto', 'auto', '10/100/1000BaseTX']}


dict_of_dicts = {'Gi0/1/0': {'STATUS': 'connect', 'VLAN': '1', 'DUPLEX': 'auto', 'SPEED': 'auto', 'TYPE':
'10/100/1000BaseTX'}, 'Gi0/1/1': {'STATUS': 'connect', 'VLAN': '1', 'DUPLEX': 'half-duplex', 'SPEED':
'100', 'TYPE': '10/100/1000BaseTX'}, 'Gi0/1/2': {'STATUS': 'notconnect', 'VLAN': '1', 'DUPLEX': 'full-
duplex', 'SPEED': '1000', 'TYPE': '10/100/1000BaseTX'},
'Gi0/1/3': {'STATUS': 'notconnect', 'VLAN': '1', 'DUPLEX': 'auto', 'SPEED': 'auto', 'TYPE':
'10/100/1000BaseTX'}}
```


Importing complex structured data

```
>>> sdm.list_of_lists
[['PORT_NAME', 'STATUS', 'VLAN', 'DUPLEX', 'SPEED', 'TYPE'], ['Gi0/1/0', 'connect', '1',
'auto', 'auto', '10/100/1000BaseTX'], ['Gi0/1/1', 'connect', '1', 'half-duplex', '100',
'10/100/1000BaseTX'], ['Gi0/1/2', 'notconnect', '1', 'full-duplex', '1000',
'10/100/1000BaseTX'], ['Gi0/1/3', 'notconnect', '1', 'auto', 'auto', '10/100/1000BaseTX']]


>>> sdm.dict_of_lists
{'Gi0/1/0': ['connect', '1', 'auto', 'auto', '10/100/1000BaseTX'], 'Gi0/1/1': ['connect', '1',
'half-duplex', '100', '10/100/1000BaseTX'], 'Gi0/1/2': ['notconnect', '1', 'full-duplex',
'1000', '10/100/1000BaseTX'], 'Gi0/1/3': ['notconnect', '1', 'auto', 'auto',
'10/100/1000BaseTX']}


>>> sdm.dict_of_dicts
{'Gi0/1/0': {'STATUS': 'connect', 'VLAN': '1', 'DUPLEX': 'auto', 'SPEED': 'auto', 'TYPE':
'10/100/1000BaseTX'}, 'Gi0/1/1': {'STATUS': 'connect', 'VLAN': '1', 'DUPLEX': 'half-duplex',
'SPEED': '100', 'TYPE': '10/100/1000BaseTX'}, 'Gi0/1/2': {'STATUS': 'notconnect', 'VLAN':
'1', 'DUPLEX': 'full-duplex', 'SPEED': '1000', 'TYPE': '10/100/1000BaseTX'}, 'Gi0/1/3':
{'STATUS': 'notconnect', 'VLAN': '1', 'DUPLEX': 'auto', 'SPEED': 'auto', 'TYPE':
'10/100/1000BaseTX'}}
```



Writing to a CSV file

```
~$ cat write_csv.py
#! /usr/bin/env python3
""" CSV Write """

import csv

header = ["name", "model", "year"]
data = [
    ["toyota", "corolla", "2016"],
    ["ford", "escort", "2016"],
    ["nissan", "yaris", "2019"],
    ["seat", "leon", "2020"],
    ["volvo", "xc60", "2021"],
]
cars_csv = "cars.csv"

# // Open CSV file as a file-handle //
with open(cars_csv, mode="w", encoding="utf8") as fh:
    writer = csv.writer(fh)
    writer.writerow(header)
    writer.writerows(data)

# // End //

```

```
~$ ./write_csv.py
```

```
~$ cat cars.csv
name,model,year
toyota,corolla,2016
ford,escort,2016
nissan,yaris,2019
seat,leon,2020
volvo,xc60,2021
```

	A	B	C
1	name	model	year
2	toyota	corolla	2016
3	ford	escort	2016
4	nissan	yaris	2019
5	seat	leon	2020
6	volvo	xc60	2021

Reading from CSV files

```
~$ cat orders.csv
Date,Region,Rep,Item,Units,Unit Cost,Total
10/5/2021,East,Ryan,Notebook,91,2.99,272.09
10/5/2021,South,Stephens,Folder,46,18.99,873.54
11/5/2021,South,Morgan,Notebook,32,4.99,159.68
11/5/2021,South,Gill,Biro,23,18.99,436.77
11/5/2021,West,Mitchell,Notebook,58,2.99,173.42
11/5/2021,East,Ryan,Folder,56,4.99,279.44
11/5/2021,South,Hogan,Notebook,71,2.99,212.29
11/5/2021,North,Smyth,Notebook,86,4.99,429.14
11/5/2021,West,Scully,Notebook,34,2.99,101.66
11/5/2021,East,Ryan,Folder,62,8.99,557.38
12/5/2021,South,Morgan,Notebook,86,4.99,429.14
12/5/2021,East,Howard,Folder,25,2.99,74.75
12/5/2021,North,Scully,Folder,33,18.99,626.67
12/5/2021,East,Ryan,Notebook,31,4.99,154.69
12/5/2021,South,Smith,Locker,4,435,1740
```

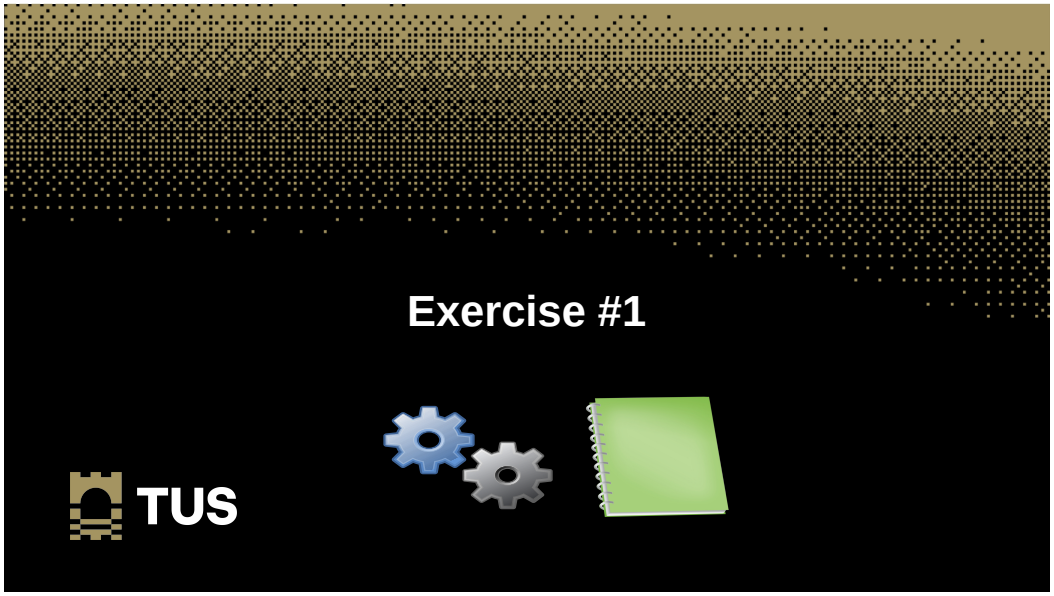
	A	B	C	D	E	F	G
1	Date	Region	Rep	Item	Units	Unit Cost	Total
2	10/5/2021	East	Ryan	Notebook	91	2.99	272.09
3	10/5/2021	South	Stephens	Folder	46	18.99	873.54
4	11/5/2021	South	Morgan	Notebook	32	4.99	159.68
5	11/5/2021	South	Gill	Biro	23	18.99	436.77
6	11/5/2021	West	Mitchell	Notebook	58	2.99	173.42
7	11/5/2021	East	Ryan	Folder	56	4.99	279.44
8	11/5/2021	South	Hogan	Notebook	71	2.99	212.29
9	11/5/2021	North	Smyth	Notebook	86	4.99	429.14
10	11/5/2021	West	Scully	Notebook	34	2.99	101.66
11	11/5/2021	East	Ryan	Folder	62	8.99	557.38
12	12/5/2021	South	Morgan	Notebook	86	4.99	429.14
13	12/5/2021	East	Howard	Folder	25	2.99	74.75
14	12/5/2021	North	Scully	Folder	33	18.99	626.67
15	12/5/2021	East	Ryan	Notebook	31	4.99	154.69
16	12/5/2021	South	Smith	Locker	4	435	1740

Reading from CSV files

```
~$ cat read_csv.py
#! /usr/bin/env python3
""" CSV Read """

import csv

header = list()
data = list()
orders_csv = "orders.csv"

# // Open CSV file as a file-handle //
with open(orders_csv, mode="r", encoding="utf8") as fh:
    csv_read = list(csv.reader(fh))
    header = csv_read[0]
    data = csv_read[1:]

print(header)
print()
[print(x) for x in data]


# // End //

```

Reading from CSV files

```
~$ ./read_csv.py
['Date', 'Region', 'Rep', 'Item', 'Units', 'Unit Cost', 'Total']

['10/5/2021', 'East', 'Ryan', 'Notebook', '91', '2.99', '272.09']
['10/5/2021', 'South', 'Stephens', 'Folder', '46', '18.99', '873.54']
['11/5/2021', 'South', 'Morgan', 'Notebook', '32', '4.99', '159.68']
['11/5/2021', 'South', 'Gill', 'Biro', '23', '18.99', '436.77']
['11/5/2021', 'West', 'Mitchell', 'Notebook', '58', '2.99', '173.42']
['11/5/2021', 'East', 'Ryan', 'Folder', '56', '4.99', '279.44']
['11/5/2021', 'South', 'Hogan', 'Notebook', '71', '2.99', '212.29']
['11/5/2021', 'North', 'Smyth', 'Notebook', '86', '4.99', '429.14']
['11/5/2021', 'West', 'Scully', 'Notebook', '34', '2.99', '101.66']
['11/5/2021', 'East', 'Ryan', 'Folder', '62', '8.99', '557.38']
['12/5/2021', 'South', 'Morgan', 'Notebook', '86', '4.99', '429.14']
['12/5/2021', 'East', 'Howard', 'Folder', '25', '2.99', '74.75']
['12/5/2021', 'North', 'Scully', 'Folder', '33', '18.99', '626.67']
['12/5/2021', 'East', 'Ryan', 'Notebook', '31', '4.99', '154.69']
['12/5/2021', 'South', 'Smith', 'Locker', '4', '435', '1740']
```

Exercise #1

- Write a program called "**`exercise_5.1.py`**".
- Read the CSV file, **`orders.csv`**, adding the following data to the retrieved data and writing a new **`.csv`** file, **`orders2.csv`**, with the complete dataset.

```
['14/5/2021', 'West', 'Duggan', 'Stapler', '5', '15', '75']
```

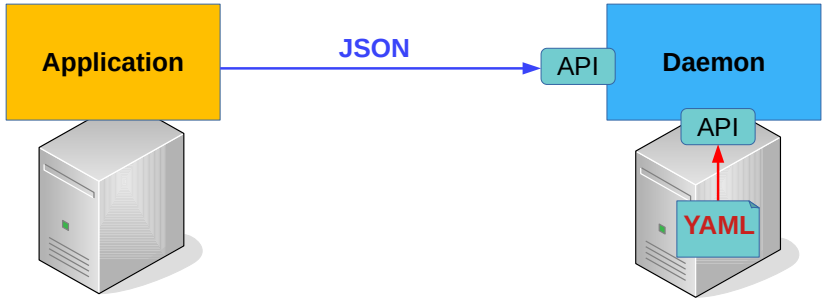
- The new file should look like this:

```
~$ tail -5 orders2.csv
12/5/2021,East,Howard,Folder,25,2.99,74.75
12/5/2021,North,Scully,Folder,33,18.99,626.67
12/5/2021,East,Ryan,Notebook,31,4.99,154.69
12/5/2021,South,Smith,Locker,4,435,1740
14/5/2021,West,Duggan,Stapler,5,15,75
```


Serialisation Protocols

- Serialisation protocols converting data objects that exist within complex data structures into a byte stream for storage or transfer.
- There are a number of data serialisation protocols and their use varies depending upon factors such as data complexity, need for human readability, speed and storage space constraints.
  - JavaScript Object Notation (JSON)
  - YAML Ain't Markup Language (YAML)
  - Extensible Markup Language (XML)

Serialisation Protocols



Processing YAML

```
~$ cat yaml_start.py
#! /usr/bin/env python3
""" Yaml start """

import sys
import yaml

# Variables
file = "data.yaml"
file2 = "data2.yaml"


print()
# // Open a yaml file for reading //
with open(file, mode="r") as fh:
    head_ = fh.readline()
    yaml_in = yaml.load(fh, Loader=yaml.FullLoader)
    print(f"{type(yaml_in)}\n{yaml_in}")
    print()

print("\n---------------\n")

while True:
    input("Press 'Enter' to continue")
    print()

    # // Open a multi-doc yaml file for reading //
    with open(file2, mode="r") as fh:
        head_ = fh.readline()
        yaml_in2 = yaml.load_all(fh, Loader=yaml.FullLoader)
        print(type(yaml_in2), yaml_in2)
        for doc in yaml_in2:
            print(doc)
            print("\n---------------\n")

```

```
~$ cat data.yaml
---

 # List
 - list 1
 - list 2
 - list 3
 - list 4

 ...

~$ cat data2.yaml
---

 # List
 - list 1
 - list 2
 - list 3
 - list 4

 ...
```

Processing YAML - lists

```
~$ ./yaml_start.py

<class 'list'>
['list 1', 'list 2', 'list 3', 'list 4']

---------------
Press 'Enter' to continue


<class 'generator'> <generator object load_all at 0x7f2d9befc120>
['list 1', 'list 2', 'list 3', 'list 4']
---------------
Press 'Enter' to continue
```


Processing YAML – A basic dictionary

```
~$ cat <<EOM>> data2.yaml
---

# A basic dictionary

dict1: one
dict2: two
dict3: three
dict4: four

EOM



Press 'Enter' to continue


{'dict1': 'one', 'dict2': 'two', 'dict3': 'three', 'dict4': 'four'}
---------------
Press 'Enter' to continue
```


Processing YAML – A more complex dictionary

```
~$ cat <<EOM>> data2.yaml
---

# More complex dictionary

dict_:
  dict1: one
  dict2: two
  dict3: three
  dict4: four

EOM



Press 'Enter' to continue

{'dict_': {'dict1': 'one', 'dict2': 'two', 'dict3': 'three', 'dict4': 'four', 'dict5': None}}
---------------
Press 'Enter' to continue
```


Processing YAML – Dictionary in a List & `null`

```
~$ cat <<EOM>> data2.yaml
---

# Dictionary in a list

- dict_:
    dict1: one
    dict2: two
    dict3: three
    dict4: four
    dict5: null

EOM



Press 'Enter' to continue

[{'dict_': {'dict1': 'one', 'dict2': 'two', 'dict3': 'three', 'dict4': 'four', 'dict5': None}}]
---------------
Press 'Enter' to continue
```

Processing YAML – List in a Dictionary

```
~$ cat <<EOM>> data2.yaml
---
# Lists in a dictionary

dict_:
  dict1:
    - sometimes: true
    - othertimes: false
    - eventimes: yes
    - goodtimes: no
  dict2: two
  dict3: three
  dict4: four
  dict5: null

EOM


Press 'Enter' to continue

{'dict_': {'dict1': [{'sometimes': True}, {'othertimes': False}, {'eventimes': True},
{'goodtimes': False}], 'dict2': 'two', 'dict3': 'three', 'dict4': 'four', 'dict5': None}}
---------------
Press 'Enter' to continue
```


Processing YAML - Multiline

```
~$ cat <<EOM>> data2.yaml
---
# Other stuff

- dict1: |
    multi line can
    look exactly
    like this.

- dict2: >
    This looks easier
    to read here within
    the YAML file.

EOM


Press 'Enter' to continue

[{'dict1': 'multi line can \nlook exactly \nlike this.\n'}, {'dict2': 'This looks easier to
read here within the YAML file.\n'}]
---------------
Press 'Enter' to continue
```


Processing YAML

```
~$ cat /yaml_processing.py
#! /usr/bin/env python3
""" Yaml processing """

import sys
import yaml
from pprint import pprint

# Variables
file = "network.yaml"
new_file = "network2.yaml"

# // Open a yaml file for reading //
with open(file, mode="r") as fh:
    head_ = fh.readline()
    net_list = yaml.safe_load(fh)

print(head_)
print(type(net_list), net_list)
print()
pprint(net_list)
print()

# // Print the 'ens160' interface //
print(f"ens160: {net_list['network']['ethernets']['ens160']}\n")

# Add static address on 'ens160'
#   Address: '10.10.10.100/16
#   Nameserver:'8.8.8.8'


new_addr = {"addresses": ["10.10.10.100/16"],
            "nameservers": {"addresses": ["8.8.8.8"]}}

# // Update the 'net_list' dictionary //
net_list["network"]["ethernets"]["ens160"] = new_addr

print(f"ens160: {net_list['network']['ethernets']['ens160']}\n")

# // Write a new network yaml file //
head2_ = "# New network2.yaml file"
print(f"{head2_}\n")
with open(new_file, "w") as fh:
    fh.write(f"{head2_}\n")
    fh.write("\n")
    yaml.dump(net_list, fh, default_flow_style=False)

# // End //
sys.exit(0)
```


Processing YAML

```
~$ ./yaml_processing.py
# network.yaml file

<class 'dict'> {'network': {'ethernets': {'ens160': {'dhcp4': True}, 'ens192': {'addresses': ['192.168.0.2/24'],
'nameservers': {'addresses': ['8.8.8.8']}}, 'ens224': {'addresses': ['192.168.1.2/24'], 'nameservers':
{'addresses': ['8.8.8.8']}, 'routes': [{'to': '192.168.2.0/24', 'via': '192.168.1.3'}]}}, 'version': 2,
'renderer': 'networkd'}}

{'network': {'ethernets': {'ens160': {'dhcp4': True},
                           'ens192': {'addresses': ['192.168.0.2/24'],
                                      'nameservers': {'addresses': ['8.8.8.8']}},
                           'ens224': {'addresses': ['192.168.1.2/24'],
                                      'nameservers': {'addresses': ['8.8.8.8']},
                                      'routes': [{'to': '192.168.2.0/24',
                                                  'via': '192.168.1.3'}]}},
            'renderer': 'networkd',
            'version': 2}}

ens160: {'dhcp4': True}

ens160: {'addresses': ['10.10.10.100/16'], 'nameservers': {'addresses': ['8.8.8.8']}}

# New network2.yaml file
```

JavaScript Object Notation (JSON)

Processing JSON

```
~$ cat data_set.json
{
    "firstName": "Tom",
    "lastName": "Clarke",
    "hobbies": ["sailing", "swimming", "hillwalking"],
    "age": 40,
    "children": [
        {
            "firstName": "Cian",
            "age": 12
        },
        {
            "firstName": "Conor",
            "age": 13
        }
    ]
}
```

```
~$ cat data_set2.json
{
  "firstName": "Tom",
  "lastName": "Clarke",
  "hobbies": [
    "sailing",
    "swimming",
    "hillwalking",
    "rugby"
  ],
  "age": 40,
  "children": [
    {
      "firstName": "Cian",
      "age": 12
    },
    {
      "firstName": "Conor",
      "age": 13
    },
    {
      "firstName": "Orla",
      "age": 1
    }
  ]
}
```


Processing JSON

```
~$ cat json_processing.py
#! /usr/bin/env python3
""" JSON processing """

import json
import pprint

# Variables //
file = "data_set.json"
new_file = "data_set2.json"

# // De-serialise data from the file (read it) //
print(f"De-serialising data from '{file}'")
with open(file, mode="r") as fh:
    data = json.load(fh)

print(f"\n{type(data)}, {data}\n")

# // Add 'rugby' as a hobby //
print("Adding 'rugby' as a new hobby")
data["hobbies"].append("rugby")

# // Add a new child //
print("Adding a new child called 'Orla'")
_ = {"firstName": "Orla", "age": 1}
data["children"].append(_)

# // Serialise data to the file (write it) //
print(f"Serialising data to '{new_file}'\n")
with open(new_file, mode="w") as fh2:
    json.dump(data, fh2, indent=("  "))
    # json.dump(data, fh2)
```

```
~$ ./json_processing.py
De-serialising data from 'data_set.json'

<class 'dict'>, {'firstName': 'Tom',
'lastName': 'Clarke', 'hobbies':
['sailing', 'swimming', 'hillwalking'],
'age': 40, 'children': [{'firstName':
'Cian', 'age': 12}, {'firstName':'Conor',
'age': 13}]}

Adding 'rugby' as a new hobby
Adding a new child called 'Orla'
Serialising data to 'data_set2.json'
```


Processing JSON

- **`json.dump()`**
  - method can be used for writing to JSON file.
- **`json.dumps()`**
  - method can convert a Python object into a JSON string.

```
>>> import json
>>> dict_ = {'one': 1, 'two': 2, 'three': 3}
>>> print(type(dict_), dict_)
<class 'dict'> {'one': 1, 'two': 2, 'three': 3}

>>> json_obj = json.dumps(dict_)
>>> print(type(json_obj), json_obj)
<class 'str'> {"one": 1, "two": 2, "three": 3}
```

Exercise #2

Exercise #2

- Add a shebang line and a document string "**`Exercise #5.2 in Python3`**".
- Import both **`yaml`** and **`json`** modules.
- Read from the **`network.yaml`** file already used.
- Print each interface and the IP address on it.
- Serialise the data to JSON format into a file called **`network.json`**.
- Output from the file should look like this:

```
~$ exercise_5.2.py
Reading the 'network.yaml' YAML file
ens192: 192.168.0.2/24
ens224: 192.168.1.2/24
Serialising data to 'network.json'
```


Learning Objectives

- Testing Files and Directories ✓
- Writing to and Deleting files ✓
- Reading from files ✓
- Importing Structured Data ✓
- Interact with Spreadsheets ✓
- Apply Serialisation Protocols; YAML, JSON ✓

TUS
Ollscoil Teicneolaíochta na Sionainne
Lár Tíre, An tIarthar Láir
Technological University of the Shannon:
Midlands Midwest

EUR ING Dr Diarmuid Ó Briain
Innealtóir Cairte agus
Léachtóir Sinsearach

E diarmuid.obriain@tus.ie | W tus.ie
Campas Maoilis, Páirc Maoilis,
Luimneach, V94 EC5T, Éire

CEng, FIEI
CISSP®

Thank you

TUS