NORTHERN ARIZONA UNIVERSITY

School of Informatics, Computing, and Cyber Systems

INF502

Lecture: Python: Files, Exceptions, and some Modules

Dr. Igor Steinmacher

e-mail: Igor.Steinmacher@nau.edu

Twitter: @igorsteinmacher

Previously…

- Language Basics: format, variables
- Conditional
- Functions
- Collections
- Loops

From Now On

- Input/Output (I/O)
  - **Files**
- Errors and Exception
- Modules

Dealing with Files

- A file in Python serves as a link to an actual file from the computer
- Read from/write to a file:

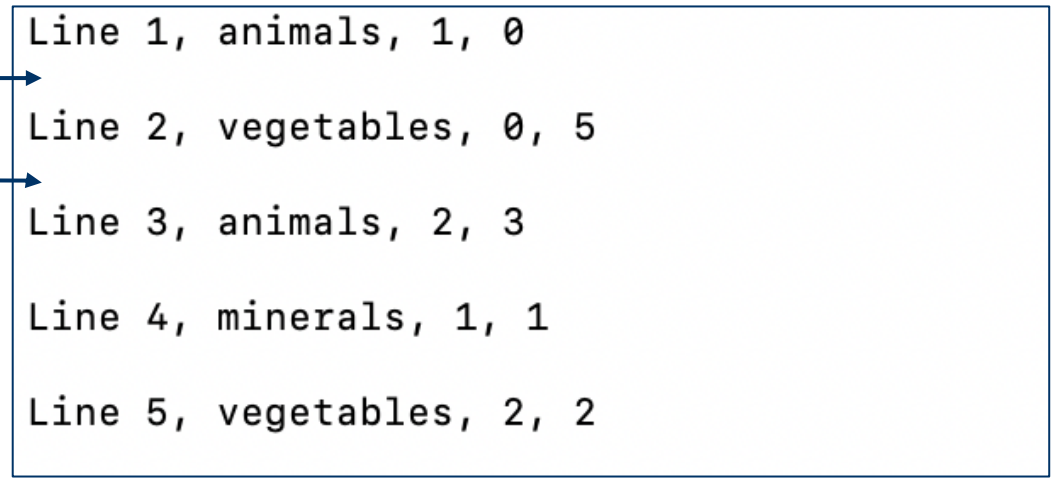
- Reading from a file:

file_handle = open('data.csv', 'r')	Open the file 'csv' in read mode
content = file_handle.read()	Read whole file content into a string
content = file_handle.read(N)	Reads N bytes (N >= 1) into a string
line = file_handle.readline ()	Read one line from the file
lines = file_handle.readlines()	Returns a list of line strings
file_handle.close()	Close the file

READING FROM A FILE

```
file_handler = open('lines.csv', 'r')
for line in file_handler.readlines():
    print (line)
file_handler.close()
```

```
['Line 1, animals, 1, 0\n', 'Line 2, vegetables, 0, 5\n', 'Line 3, animals, 2, 3\n', 'Li
ne 4, minerals, 1, 1\n', 'Line 5, vegetables, 2, 2\n']
```



READING FROM A FILE

```
file_handler = open('lines.csv', 'r')
for line in file_handler.readlines():
    print (line.rstrip('\n'))
file_handler.close()
```

```
['Line 1, animals, 1, 0\n', 'Line 2, vegetables, 0, 5\n', 'Line 3, animals, 2, 3\n', 'Li
ne 4, minerals, 1, 1\n', 'Line 5, vegetables, 2, 2\n']
```

```
[...
Line 1, animals, 1, 0
Line 2, vegetables, 0, 5
Line 3, animals, 2, 3
Line 4, minerals, 1, 1
Line_5, vegetables, 2, 2
```

READING FROM A FILE

```
file_handler = open('lines.csv', 'r')
for line in file_handler.readlines():
    line = line.rstrip('\n')
    field = line.split(',') #break the string into a list (split - comma)
    print (field)
file_handler.close()
```

```
['Line 1', ' animals', ' 1', ' 0']
['Line 2', ' vegetables', ' 0', ' 5']
['Line 3', ' animals', ' 2', ' 3']
['Line 4', ' minerals', ' 1', ' 1']
['Line 5', ' vegetables', ' 2', ' 2']
```

Writing to a file

file_handle = open(‘data.csv’, ‘w’)	Open the file ‘data.csv’ in write mode
content = file_handle.write(S)	Write string S to a file
content = file_handle.writelines(L)	Write the strings in the list L to a file
file_handle.close()	Close the file

Writing to a file

```
file_handler = open('other.csv', 'w')
list_1 = ('banana', 'carrot', 'avocado', 'orange', 'grapefruit')
file_handler.writelist(list_1)
file_handler.close()
```

```
file_handler = open('other.csv', 'w')
list_1 = ('banana', 'carrot', 'avocado', 'orange', 'grapefruit’)
for item in list_1:
   file_handler.write(item + '\n')
file_handler.close()
```

File Open Modes

Character	Meaning
'r'	open for reading (default)
'w'	open for writing, truncating the file first
'x'	create a new file and open it for writing
'a'	open for writing, appending to the end of the file if it exists
'b'	binary mode
't'	text mode (default)
'+'	open a disk file for updating (reading and writing)
'U'	universal newline mode (deprecated)

HANDS ON: 10 MINUTES

- Create a file called grades.csv, with the following information:

  John,9,9,5,9

  Peter,7,4.3,6.7

  Junior,7,3,2,4

  Anthony,3,3,7.5,10

- Now write an algorithm that:
  1. **For each line in the file, calculates the mean (for each person);**
  2. **Defines if the person Passed or Failed (status)**
  3. **Write a report (file named report.txt) in which each line needs to inform:**

     <NAME> <STATUS> (<MEAN>), for example:

     John Passed (8.0)

Erros And Exceptions

- *Syntax errors* and *exceptions*
  - ***Syntax Errors* occur when the *grammar* of a Python statement is incorrect**

```
>>> x = 2
>>> if (x > 0)
  File "<stdin>", line 1
    if (x > 0)
              ^
SyntaxError: invalid syntax
>>>
```

  - **Exceptions are errors that happen in execution time, even with correct Syntax**

```
>>> list1 = (1,2,3,4,5)
>>> list1[8]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
IndexError: tuple index out of range
>>>
```

Errors And Exceptions

Error	Description/example
IOError	e.g. file not found
IndexError	an attempt to access a sequence (such as a list with an index out of range)
TypeError	an operation or function applied to an object of inappropriate type
NameError	a variable name is not recognised
ValueError	an operation or function receives an argument of the right type but an inappropriate value
ZeroDivisionError	a specific type of ValueError raised when an attempt is made to divide by zero.

Exception Hunting

- Knowing the potential exceptions is key… Experience (bad ones) help you
- Exceptions can be 'caught' and handled in Python script:
  - **try … catch … finally**

```
try:
   # <do something that might fail>
except (<exception1>, <exception2>, ...):
   # <something went wrong: deal with it>
else:
  # <what to do if no exception>
finally:
   # <statements here are always executed>
```

Exception Hunting

- Knowing the potential exceptions is key… Experience (bad ones) help you
- Exceptions can be ‘caught’ and handled in Python script:
  - **try … catch … finally**

```
filename = input("Enter a file name: ")
try:
    f = open(filename, "r")
except FileNotFoundError:
    print("There is no file named", filename)
```

Exception Hunting

```
print ("Are you authorized to drink in AZ?")
try:
  age = input ("Type your age: ")
  age = int(age)

except (ValueError):
  print ("The value typed is not an integer")

else:
  if (age < 21):
    print ("You cannot drink")
  else:
    print ("You can drink")
    age = int(age)
```

RAISING EXCEPTIONS

```
def canDrink (age):
  if (age < 0):
    #a new ValueError will be raised to who called canDrink
    my_error = ValueError("{0} is not a valid age ". format(age))
    raise my_error
  result = True if (age >= 21) else False
  return result
```

HANDS ON!

Write a function named **readposint** that uses the **input** dialog to prompt the user for a positive integer and then checks the input to confirm that it meets the requirements. It should be able to handle inputs that cannot be converted to int, as well as negative ints, and edge cases (e.g. when the user closes the dialog, or does not enter anything at all.)

Some Python Modules

- Longer programs can be split up into separate files
- Each file, with functions and variables, can be considered a *module*
- Modules are *imported* using the **import** statement
- The Standard Python Library consists of some modules (and packages of modules)

Some Python Modules

math	Mathematical functions
sys	System-specific parameters and functions
os	Miscellaneous operating system interfaces
random	Generate (pseudo-)random numbers
zlib	A compression library (compatible with gzip)
argparse	Command-line argument parsing
urllib	Open and access resources across the internet by URL
datetime	Basic date and time types
re	Regular expressions

MATH MODULE

- sqrt(x)
- exp(x)
- log(x)
- log(x, base)
- log10(x)
- sin(x), etc.
- asin(x), etc.
- sinh(x), etc.
- hypot(x, y)
- pi
- e

SYS MODULE

- System methods and constants, of which the most useful are:
  - **argv** **The list of command line arguments passed to a Python script.**
    - argv[0] is the script name. e.g.
    - $ python my_script.py hello 4
      - sys.argv[0] = ‘my_script.py’
      - sys.argv[1] = ‘hello’
      - sys.argv[2] = ‘4’
  - **exit([n])**
    - Exit from Python. If not provided – n defaults to 0, indicating normal termination.
    - Different non-zero values are used to indicate to the shell various errors.

SYS MODULE

```
import sys
try:
    x = float(sys.argv[1])
except (IndexError, ValueError):
    sys.exit("I need a number, please.")
try:
    rx = 1./x
except ZeroDivisionError:
    sys.exit("You can't divide by zero!")
print(rx)
```

OS MODULE

- Miscellaneous operating system interfaces
  - **path: Manipulate file and directory names**
  - **environ: A mapping object representing the string environment. *e.g.* os.environ['HOME'] is the pathname of your home directory (on some systems)**
  - **remove: Delete a file**
  - **rename: Rename a file**
  - **stat: get information about file read/write permissions, last time of modification, etc.**
  - **listdir : Return a list of the entries in a directory**

OS MODULE

```
>>> import os
>>> HOME = os.environ['HOME']
>>> print(os.listdir(HOME))
['.bash_history', '.bash_profile', 'Desktop', 'Documents, 'Downloads',
'Library', 'research', 'teaching', ... ]
```