Module 1

**How to run a Python Script**

Shebang line (first line of code): #!/usr/bin/env python3

Change mode to executable in terminal:

Chmod +x name.py

Then executing it

./name.py

(. Means the current directory)

**Your own python modules**

A single .py file can be a module. We have to import the module, the use . operator to access things inside the module. A big module is one with the module name as the directory name, then with a bunch of .py files as submodules. It will contain an \_\_init\_\_.py file, which tells the compiler to treat the directory as a module.

Module 2

**Programming with Files**

**Reading files**

To open a file use file = open(“spider.txt”)

To print one line use Print(file.readline())

Use Print(file.read()) to read from the current position till the end

After that use file.close() to close the file

Use the following to automatically close

With open(“spider.txt”) as file:

Print(file.readline())

**Iterating through files**

To make line uppercase, use the following

With open(“spider.txt”) as file:

For line in file:

Print(line.upper())

The above has lines in between due to the file having a newline character and the print adding another newline. We can remove this using strip()

Print(line.upper().strip())

We can also read the file lines into a list then sort it using readlines()

File = open(“spider.txt”)

Lines = file.readlines()

File.close()

Lines.sort()

Print(lines)

**Writing files**

To write to a file use the following

With open(‘novel.txt’, ‘w’) as file:

File.write(“some line”)

This returns the number of characters written

If the mode is “w” , if the file doesn’t exist python will create it, if it exists, then the contents will be overwritten

Different modes are

“w” write (deletes the content)

“r” read only, default, no need to pass

“a” append

“r+” read write mode

**Working with files**

To delete a file use os.remove(“filename.txt”) (if file not exit, error is generated)

Rename a file using os.rename(“old\_name.txt”, “new\_name.txt”) (if file not exit, error is generated)

Check if a file exists using os.path.exists(“filename.txt”) true if exists

**More file information**

To check how big a file is use Os.path.getsize(“filename.txt”) returns filesize in bytes

Check when the file is last modified, use os.path.getmtime(“filename.txt”) returns a timestamp, seconds since jan 1970

Get time from timestamp using following code snippet

Import datetime

Timestamp = os.path.getmtime(“filename.txt”)

Datetime.datetime.fromtimestamp(timestamp) (Gets the date)

To use absolute path of file use Os.path.abspath(‘spider.txt’) returns the absolute path (useful to get the full path irrespective of os)

**Directories**

Get working directory

Print(os.getcwd())

Create dir

Os.mkdir(“new\_dir”)

Change dir to new\_dir

Os.chdir(“new\_dir”)

Remove dir

Os.mkdir(“newer\_dir”)

Os.rmdir(“newer\_dir”) – will work only if dir is empty

Contents in a dir

Os.listdir

Returns a list of strings, can’t understand if the list contents are dir or files

Eg: os.listdir(“website”)

[‘images’, ‘index.html’, ‘favicon.ico’]

Use os.path.join(dir, name) to join directory with file name ie, full name (helps in os independence, linux, macos uses /, whereas windows uses \ , join takes care of that

Use os.path.isdir(fullname) to check if fullname is a dir or file

**CSV files**

**Reading csv files**

Import csv

Opening a csv file

F = open(“csv\_file.txt”)

Csv\_f = csv.reader(f)

For row in csv\_f:

Name, phone, role = row

Print(“Name: {}, Phone: {}, Role: {}”.format(name, phone, role))

f.close()

Result

Name: A, Phone: 123, Role: Sys Admin

**Generating csv files**

Eg:

Hosts = [[“workstation.local”, “192.168.25.46”], [“webserver.cloud”, “10.2.5.6”]]

With open(‘hosts.csv, ‘w’) as hosts\_csv:

Writer = csv.writer(hosts\_csv)

Writer.writerows(hosts) (writerow writes one row)

Ubuntu: cat host.csv (seeing the file outside python in ubuntu)

**Reading and writing csv files with dictionary**

When reading csv files with more columns, usually the first row contains the column names

we use dictreader to read it as a dictionary

eg: cat software.csv

name,version,status,users

MailTree,5.34,production,324

CalDoor,1.25,betta,22

with open('software.csv') as software:

reader = csv.DictReader(software)

for row in reader:

print('{} has {} users").format(row['name'], row['users']))

result: MailTree has 324 users

CalDoor has 22 users

Also dictwriter is used to write as a dictionary

eg

users = [{'name': 'abc', username': 'ab', 'dept': 'it'},

{'name': 'xyz', 'username': 'xy', 'dept':'ui'}]

keys = ['name', 'username', 'dept']

with open('by\_dept.csv', 'w') as by\_dept:

writer = csv.DictWriter(by\_dept, fieldname=keys)

writer.writeheader()

writer.writerows(users)

**Module 3**

**Why use regex?**

Eg: log = “july is a good month, [1234] do u know this?”

Index = log.index(‘[‘)

Print(log(index+1: index+5])

Another way

Import re

log = “july is a good month, [1234] do u know this?”

regex = r”\[(\d+)\]”

result = re.search(regex, log)

print(result[1])

**Basic matching with grep**

Searching for the word “thon” in linux in the file /usr/share/dict/words. This file contains one word per line

Grep thon /usr/share/dict/words ( this is case sensitive)

Grep –i pyton /usr/share/dict/words (to ignore case sensitivity)

A dot matches any character

Eg: grep l.rts /usr/share/dict/words (matches ‘alerts’, ‘blurts’)

^ marks the beginning of the regex matching of a line (not string)

$ marks the end of the regex matching of a line (not string)

Eg: grep ^ fruit /usr/share/dict/words

(matches fruitcake, fruity)

Ge: grep cat$ /usr/share/dict/words

(matches bobcat, Muscat)

**Basic matching in python**

Eg: import re

Result = re.search(r”aza”, “plaza”)

Span(2, 5)

Result = re.search(r”aza”, “bazaar”)

Span(1, 4)

Result = re.search(r”aza”, “maze”)

None

Print(re.search(r”^x”, “xenon”))

Match=’x’

Print(re.search(r”p.ng”, “penguin”))

Match(peng)

Print(re.search(r”p.ng”, “clapping”))

Match(ping)

To ignore cases

Print(re.search(r”p.ng”, “Pangaea”, re.IGNORECASE)

**Wildcards and character classess**

Dot is a wildcard. They matches any other character

Character classess are written inside square brackets. They match only specified characters

Eg: for matching Python/python we use

Print(re.search(r”[Pp]ython”, “Python”))

Another eg:

Print(re.search(r”[a-z]way”, “the end of the highway”))

Match= hway

Print(re.search(r”[a-z]way”, “what a way to go”))

None

Another: print(re.search(“cloud[a-zA-Z0-9]”, “cloudy”))

Match = cloudy or cloud9

For not matching use ”^”

Eg: print(re.search(r”[^a-zA-Z]”, “this is a sentence with spaces.”))

Match = ‘ ‘

print(re.search(r”[^a-zA-Z ]”, “this is a sentence with spaces.”))

match=’.’

‘|’ matches either (like or)

Print(re.search(r”cat|dog”, “I like dogs.”))

Match=’dog’

Print(re.search(r”cat|dog”, “I like both dogs and cats.”))

Match = ‘dog’ (search works for the first match)

Print(re.findall(r”cat|dog”, “I like both dogs and cats.”))

[‘dog’, ‘cat’]

**Repetition Qualifiers**

‘.\*’ means any character repeating any number of times

Eg:

Print(re.search(r”Py.\*n”, “Pygmalion”))

Match Pygmalion

Print(re.search(r”Py.\*n”, “Python Programming”))

Match=Python Programmin (\* takes a long a possible, ie it is greedy)

Print(re.search(r”Py[a-z]\*n”, “Python Programming”))

Match=python

Print(re.search(r”Py[a-z]\*n”, “Pyn”))

Match=Pyn

‘+’ matches one or more occurrences of the character before it

Print(re.search(r”o+l+”, “goldfish”)

Match ol

Print(re.search(r”o+l+”, “woolly”)

Match ooll

‘?’ means 0 or 1 occurrence of character before it

Print(re.search(r”p?each”, to each their own”))

Match=each

Print(re.search(r”p?each”, I like peaches”))

Match=peach

**Escaping characters**

Eg:

Print(re.search(r”.com”, “welcome”))

Match=lcome

Print(re.search(r”\.com”, “welcome”))

None

Print(re.search(r”\.com”, “google.com”))

Match=.com

‘\w’ matches alphanumeric characters

Print(re.search(r”\w\*”, “this is an example”))

Match=’this’

Print(re.search(r”\w\*”, “this\_is\_an\_example”))

Match= this\_is\_an\_example

\d for digits, \s for whitespaces, \b for word boundaries

[www.regex101.com](http://www.regex101.com)

**Regex in action**

Checking for valid variable name in python

Regex = r”^[a-zA-Z\_][a-zA-Z0-9]\*$”

Re.search(regex, “\_2b”)

Checking for a standard text, ie, starts with uppercase letter, followed by at least some lowercase letters or a space, and ends with a period, question mark or exclamation point.

R”^[A-Z][a-z ]+[.|?|!]$”