**Answer Sheet**

1. **Unix Commands**

**A1.**

Directly open the image from terminal using below methods.

Write the following command in terminal.

* cd /image\_directory

go the image directory where the image file resides.

* eog sample.png

This line will open the required image file

Or you can directly write the following command.

* eog /image\_directory/sample.png

**A2.**

Use the following scp command at the terminal to solve this problem.

scp -r user\_name@ip\_address\_of\_remote:/path/to/remote/directory /path/to/local/directory

**A3.**

Write the following command at the terminal to solve these problems.

1. tail -100 [filename]
2. grep -c “ImageNet” [filename]

**A4.**

Follow the process to find disk size of a specified python package

1. pip3 show [package\_name] | grep "Location:"
2. this will return path/to/all/packages.
3. du -h path/to/all/packages.
4. last line will contain size of all packages in MB.

**A5.**

To see the python version at current session, write the following command at the terminal.

python --version

1. **Python Basic**

**B1.**

def method\_name(\*arg, \*\*args):

pass

In Python, we can pass a variable number of arguments to a function using special symbols. There are two special symbols:

1. \*args (Non-Keyword Arguments)

As in the above example we are not sure about the number of arguments that can be passed to a function. Python has \*args which allow us to pass the variable number of non-keyword arguments to function.

In the function, we should use an asterisk \* before the parameter name to pass variable length arguments. The arguments are passed as a tuple and these passed arguments make tuple inside the function with same name as the parameter excluding asterisk \*.

1. \*\*args (Keyword Arguments)

Python passes variable length non keyword argument to function using \*args but we cannot use this to pass keyword argument. For this problem Python has got a solution called \*\*args, it allows us to pass the variable length of keyword arguments to the function.

In the function, we use the double asterisk \*\* before the parameter name to denote this type of argument. The arguments are passed as a dictionary and these arguments make a dictionary inside function with name same as the parameter excluding double asterisk \*\*.

**B2.**

def iterator (start, end):

pass

**iterator (10, 10)**

* This takes values of both variables “start” and “end” to 10 in the iterator function.

**iterator (start=10, 20)**

* This throws an error. The second value expects the keyword “end” which isn’t there. It throws SyntaxError: positional argument follows keyword argument. In python3 this cannot be done. Means keyword argument should be placed after positional argument.

**iterator (10, end=10)**

* This works fine. First variable takes value 10. Then keyword argument takes value 10 to variable “end”. This works as well.

**iterator (start=10, end=20)**

* This works as well. This takes two keyword argument because the iterator function two defined variable as parameter.

See the B2.py file for more details.

**B3.**

In Python, strings cannot be modified. You need to create a new string based on the contents of an old one if you want to change a string. The “'str' object does not support item assignment” error tells **you that you are trying to modify the value of an existing string. So, Item assignment of string doesn’t work in python. It throws a type error.**

See B3.py for proof.

**B4.**

There are multiple ways to solve this problem. The best way could be to call the function as follows

name, \_ = get\_info()

Also, we can get first value from get\_info() function as its return a tuple where the first value is name as follows,

name = get\_info()[0]

see the B4.py file for more details.

**B5.**

While dividing two numbers the breaking point is when the divisor is zero. To solve this issue, we have to throw an exception when the we try to divide by zero. Python has built it method to detect this. We just to use try catch block as follows to resolve this.

try:

return number1 / number2

except ZeroDivisionError:

return "You can't divide by zero"

See B5.py file for detailed working code.

1. **Python OOP**

**C1.**

The solution of the problems is written at C1.py file.

**C2.**

The solution of this problem is written at C2.py file. The solution was done using the lambda function.

1. **Data Structure and Algorithm**

**D1.**

Detailed solution and required files can be found in D1 folder.

**D2.**

Lists are slightly faster than sets when you just want to iterate over the values.

Sets, however, are significantly faster than lists if you want to check if an item is contained within it. They can only contain unique items though.

So, in this case lookup method for set will be faster as we just to check a certain is contained within or not.

**D3.**

The solution of this problem can be found in D3.py file.

In this problem two lists were generated where we search for each value of second list if it exists in the first list. Because searching in the list the search time is higher. In worst case which is O(n) thus the execution time is higher.

To lower execution time we replaced the list by dictionary. Dictionary used Hash map to store values. Thus, searching in dictionary is way faster. That’s how we lowered the execution time.

1. **Numpy**

**E1.**

The solution of this problem is written at E1.py file.

**E2.**

The solution of this problem is written at E2.py file. “np.sum(A, axis=3)” this portion of the code solves the problem.

1. **Deep Learning**

**F1.**

The solution of this problem can be found in F1.py file. To run this code first run the mredul\_exam\_env.yml file to create the environment.

**F2.**

The solution of this problem can be found in F2.py file. To run this code first run the mredul\_exam\_env.yml file to create the environment.

There are multiple ways to solve overfitting issue in neural network. In this particular code we have changed few things those are listed below:

* Changed the optimizer to “Adam”
* Reduced learning to 1e-5
* Added a dropout layer
* Reduced to number of epochs from 100 to 75

With increased numbers of epoch, the model tends to overfit due to using CNN to find similar patterns. In this problem for softmax output the Adam optimizer works better. Adding dropout and reducing learning rate are common methods of removing overfitting issue.