**A. Unix commands**

**A1.**

**Answer:**

I will use the terminal using following command:

If you find 10.png image then you write the bellow command-

>>cd /directory/

>>eog 10.png

**This also works:**

xdg-open 10.png

**A2.**

**Answer:**

SCP command-

**scp -r file1.txt folder user@<ip\_address\_of\_user>:/home/user/Desktop**

**A3.**

**Answer:**

1. Command:

**tail -n 100 <log file> > newLogfile**

1. Command:

grep -c Imagenet [filename]

**A4.**

**Answer:**

Follow the bellow command:

1. pip show <package> will show you the location
2. we get the **Name** and **Location** to join them to get the location, finally use **du -sh** to get the package size.

**A5.**

**Answer:**

>> python --version

**B. Python Basic**

**B1.**

**Answer:**

1. \*args

The special syntax \*args in function definitions in python is used to pass a variable number of arguments to a function. It is used to pass a non-key worded, variable-length argument list.

The syntax is to use the symbol \* to take in a variable number of arguments; by convention, it is often used with the word args.

1. \*\*kwargs

The special syntax \*\*kwargs in function definitions in python is used to pass a keyworded, variable-length argument list. the double star allows us to pass through keyword arguments (and any number of them).

A keyword argument is where you provide a name to the variable as you pass it into the function.

**B2.**

**Answer:**

def iterator(start, end):

pass

iterator(10, 10) call will simply execute the function and pass

iterator(start=10, 20) will show 'SyntaxError: positional argument follows keyword argument'

iterator(10, end=10) will also pass like the first one

iterator(start=10, end=20) will also pass like the first one

When you call these methods one by one, There will be a problem with “ **iterator(start=10, 20)**”. Here we get positional argument syntax error.

**Please check code B2.py to see the error.**

**B3.**

**Answer:**

No. we cannot do these operations.

Explanation:

Strings in Python are immutable. This means that they cannot be changed. If you try to change the contents of an existing string, you’re liable to find an error that says something like “‘str’ object does not support item assignment”.

**Please check code B3.py to see the error.**

**B4.**

**Answer:**

Ignoring means assigning the values to special variable underscore(\_). We're assigning the values to underscore(\_) given not using that in future code.

So we can use bellow line code-

Name, \_ = get\_info()

**B5.**

**Answer:**

The breaking point of the method is when the function 'do\_math(x, y)' will be called by the value of 'y=0' and it will raise 'ZeroDivisionError: division by zero'

I will use try and except block to fix the issue, i.e,

def do\_math(number1, number2):

try:

return number1/number2

except:

print('Divided by Zero error')

**Please check code B5.py**

**C. Python OOP**

**C1.**

**Answer:**

**Please check code C1.py**

**C2.**

**Answer:**

**Please check code C2.py**

**D. Data structure & algorithm**

**D1.**

**Answer:**

**Please check D1.py**

**D2.**

**Answer:**

**lookup\_set** is best.

It depends on what you are intending to do with it.   
Sets are significantly faster when it comes to determining if an object is present in the set (as in x in s), but are slower than lists when it comes to iterating over their contents.

**D3.**

**Answer:**

**Please check D3.py**

**E. Numpy**

**E1.**

**Answer:**

a. A[1:3, 1:3]

b. A[:, :-1]

c. A[1:, :]

**Please check E1.py for check output.**

**E2.**

**Answer:**

**Please check code E2.py**

**F. Deep learning**

**F1.**

**Answer:**

**Please check code F1.py**

**F2.**

**Answer:**

I have worked two techniques here.

First, I am trying to reduce overfitting using basic data augmentation technique, like: rotation, horizontal flip, zoom etc. using ImageDataGenerator.

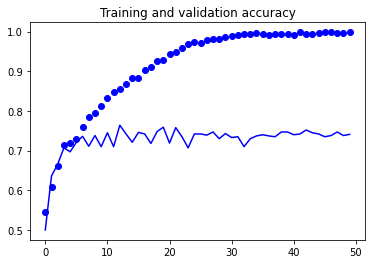
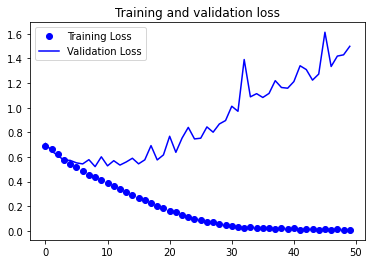
Second, I further introduce dropout layer in the model and Adam optimizer.

**Result:**

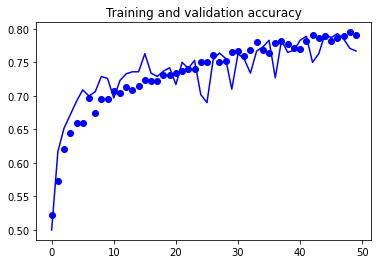
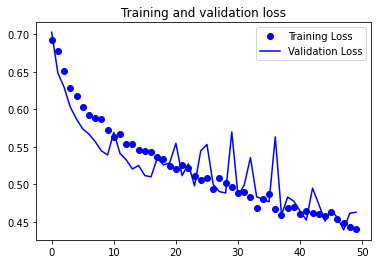
|  |  |  |
| --- | --- | --- |
| **Model Name** | **Val accuracy** | **Val Loss** |
| Baseline model | 0.7410 | 1.4982 |
| Baseline model + Basic Augmentation | 0.7930 | 0.4627 |
| **Baseline model + Basic Augmentation + Dropout + Adam** | **0.8120** | **0.4119** |

**Plot:**

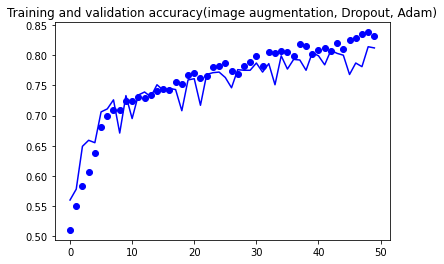
Baseline model:

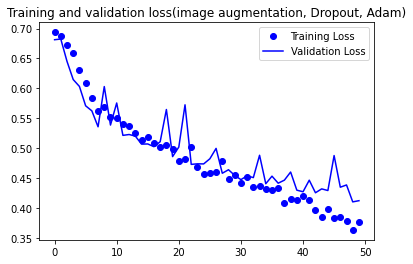


**Baseline model + Basic Augmentation:**



**Baseline model + Basic Augmentation + Dropout + Adam:**

****

****