DAY 24 OF 200 DAY'S PYTHON CHALLENGE

EXCEPTION HANDLING-2

EXCEPTION HANDLING

In Exception Handling we can add multiple exceptions in one except part or we are also add multiple except block. There are muliple types of errors or exceptions like ValueError, TypeError etc. In these types of exception of the keyword of “Exception” is also a super class exception.

1.

Python.py

```
try :
  int("abhi")
except (ValueError , TypeError) as e :
  print(e)
```

2.

Python.py

```
try :
  int("abhi")
except:
  print(“This will catch an error”)
```

EXCEPTION HANDLING

3.

Python.py

```
try :
 int("abhi")
except ZeroDivisionError as a:
 print(a)
except ValueError as e :
 print(e)
```

When we are work in multiple except block then we have a two options:

- Except a specific type of exceptions.
- Go with the super class exception but it is not good for a good developer

so we want to go with specific type of exception and if we forget a exception name then we have a multiple except block then we want to do after some except block add a super class in that case use of super class exception is good. But in this case also we remember that a except block of super class exception use at the last except block because buttom of except super class all except class never execute.

EXCEPTION HANDLING

4.

Python.py

```
try :
   with open(file , 'r') as f :
    test = f.read()
except FileNotFoundError as e :
   print("test " , e)
except Exception as e :
   print(e)
```

5.

Python.py

```
def test(file):
 try :
  with open(file , 'r') as f :
   test = f.read()
 # except Exception as e :
 #  print(e)
 except FileNotFoundError as e :
  print("test " , e)
```

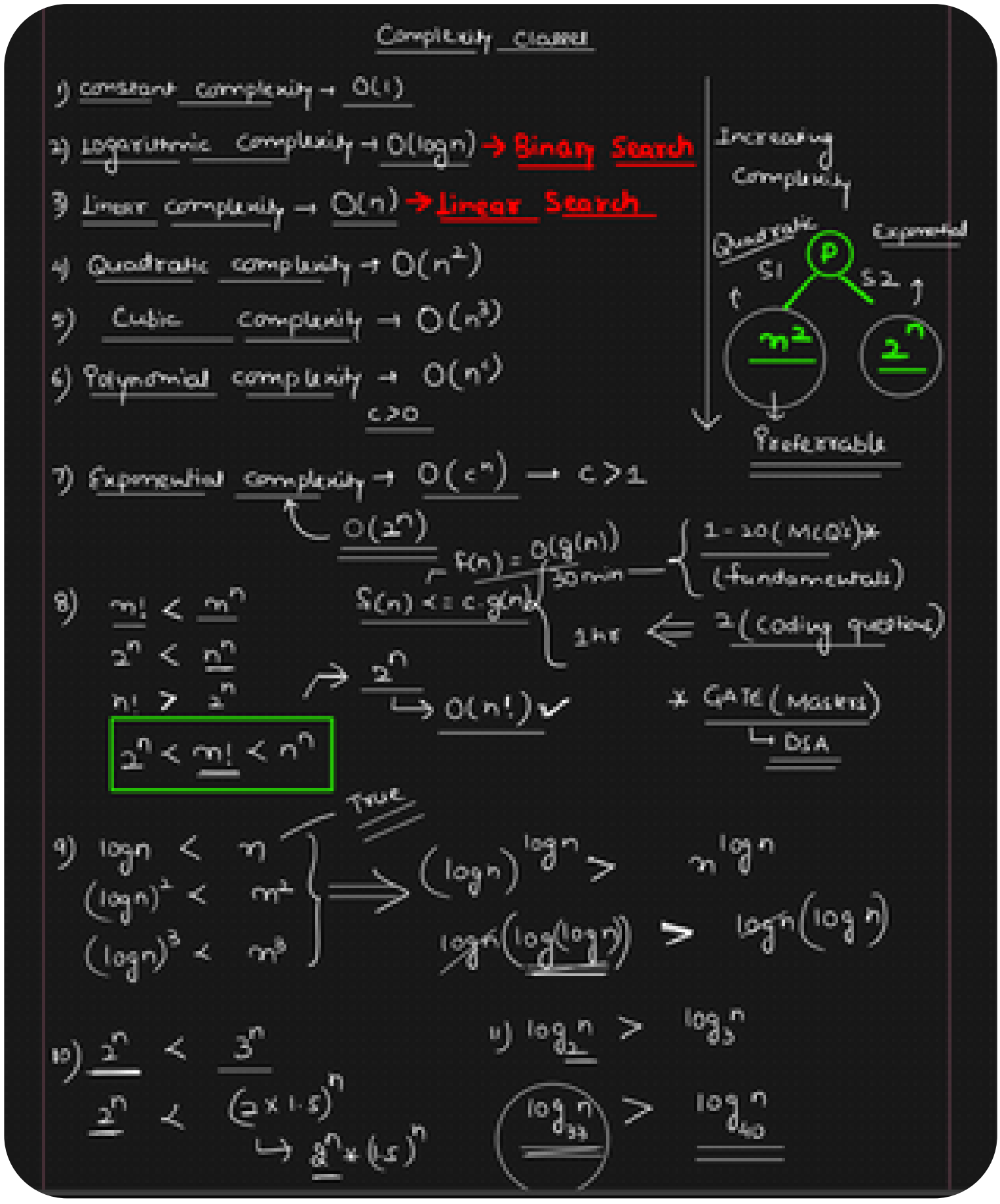
EXCEPTION HANDLING

1. SyntaxError: Raised when there is a syntax error in the code.NameError: Raised when a local or global name is not found.
2. TypeError: Raised when an operation or function is applied to an object of inappropriate type.
3. ValueError: Raised when a built-in operation or function receives an argument that has the right type but an inappropriate value.
4. ZeroDivisionError: Raised when the second operand of a division or modulo operation is zero.
5. IndexError: Raised when a sequence subscript is out of range.
6. KeyError: Raised when a dictionary key is not found.
7. FileNotFoundError: Raised when a file or directory is requested but cannot be found.
8. AttributeError: Raised when an attribute reference or assignment fails.
9. ImportError: Raised when an import statement fails to find the module definition.
10. MemoryError: Raised when an operation runs out of memory.
11. OverflowError: Raised when the result of an arithmetic operation is too large to be represented.
12. RuntimeError: Raised when an error occurs that doesn't belong to any specific built-in exception category.

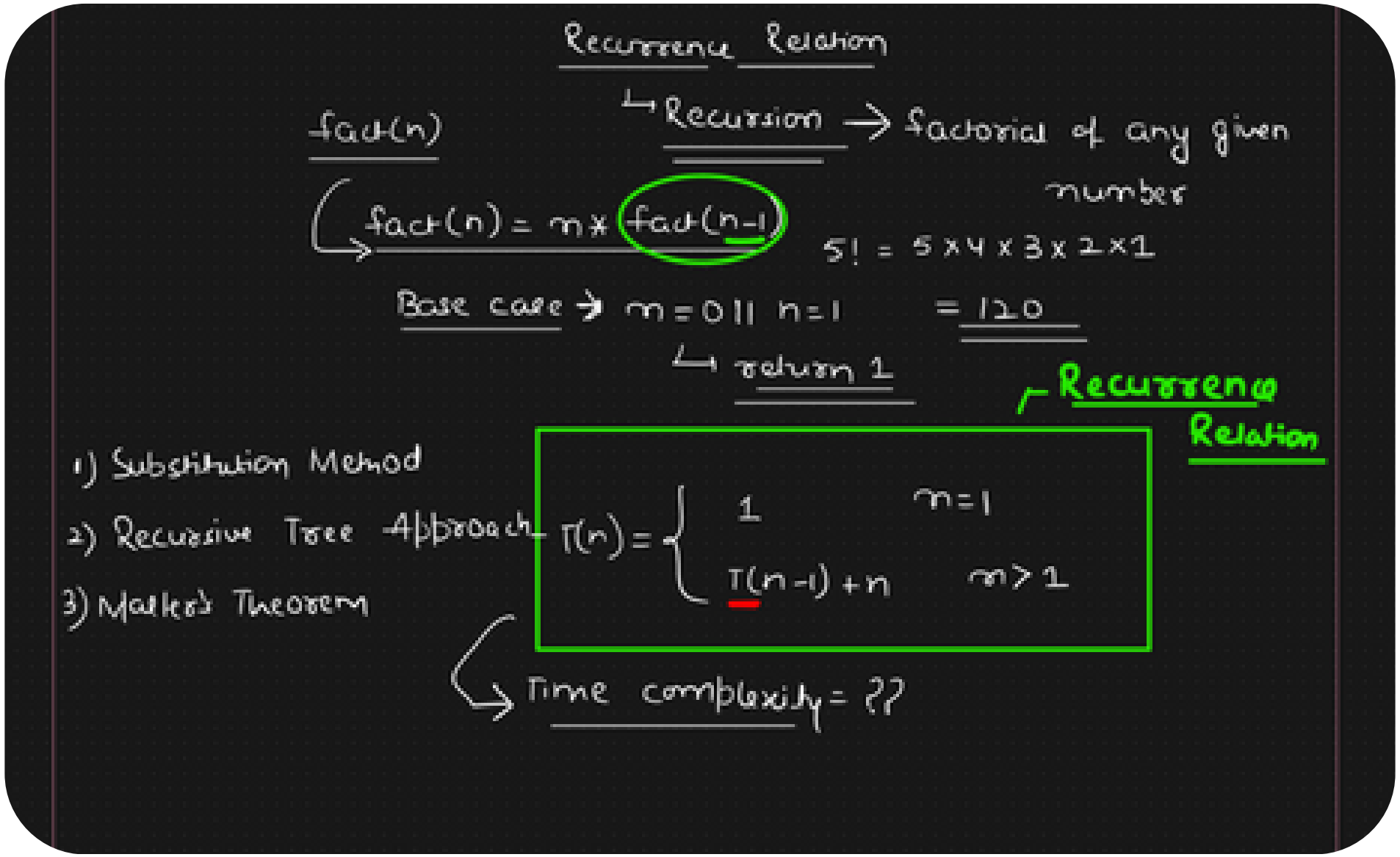
PRACTICE QUESTIONS

1. Explain why we have to use the Exception class while creating a Custom Exception.(Note: Here Exception class refers to the base class for all the exceptions.)
2. Write a python program to print Python Exception Hierarchy.
3. What errors are defined in the ArithmeticError class? Explain any two with an example.
4. Why LookupError class is used? Explain with an example KeyError and IndexError.
5. Explain ImportError. What is ModuleNotFoundError?
6. List down some best practices for exception handling in python.

COMPLEXITY IN DSA



RECURRENCE RELATION

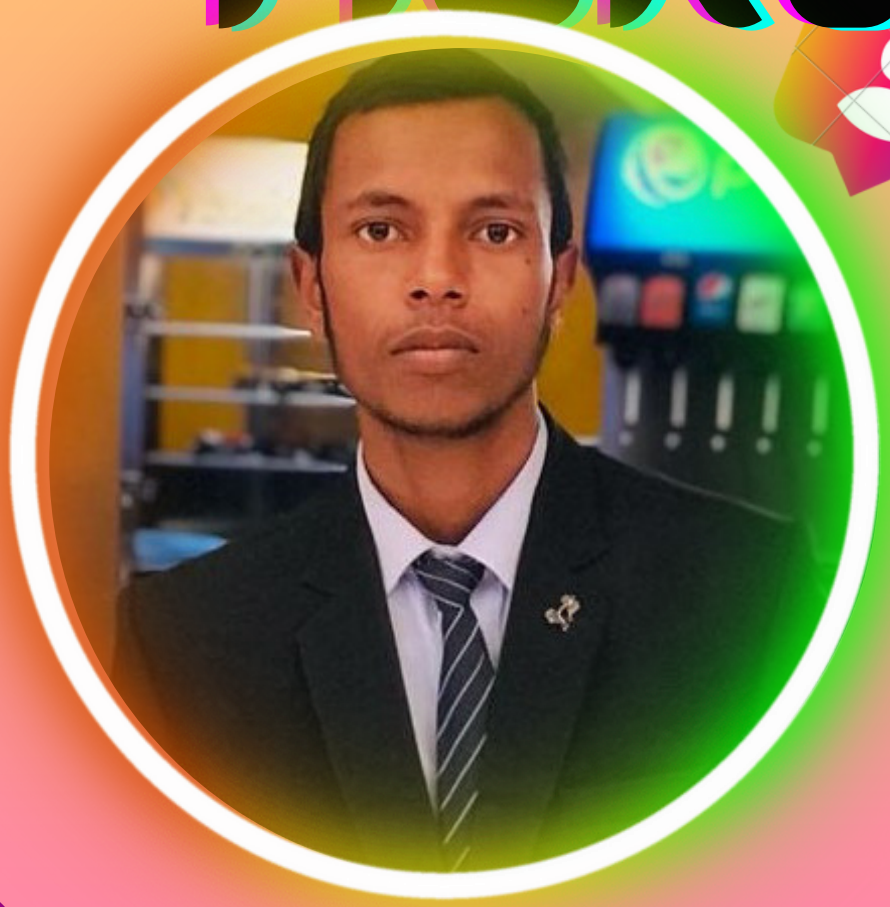


One my personal advice for a algorithm part you just check a topic name and learn in YOUTUBE because i think algo part is just logical thinking and when i was start problem solving part of DSA then i i will inform you.

Thank you! for supporting me...

Keep Learning📖, Keep Coding👩‍💻!

FOLLOW FOR MORE



FOLLOW ME