

Chapter 12 - Advanced Python 1

Exception Handling in Python

There are many built-in exceptions which are raised in Python when something goes wrong.

Exceptions in Python can be handled using a try statement. The code that handles the exception is written in the except clause.

try :

Code

→ Code which might throw Exception.

except Exception as e :

print(e)

When the exception is handled, the code flow continues without program interruption.

We can also specify the exceptions to catch like below :

try :

Code

except ZeroDivisionError :

Code

except TypeError :

code

except :

Code

→ All other exceptions are handled here.

Raising Exceptions

We can raise custom exceptions using the raise keyword in python.

try with else clause

Sometimes we want to run a piece of code when try was successful.

try :

Some code

except :

Some code

else :

Code

→ This is executed only if the try was successful

try with finally

Python offers a finally clause which ensures execution of a piece of code irrespective of the exception.

try :

Some code

except :

Some code

finally :

Some code

→ Executed regardless of error!

if `--name-- == '__main__'` in Python

`--name--` evaluates to the name of the module in Python from where the program is run

If the module is being run directly from the command line, the `--name--` is set to string `"__main__"`

Thus this behaviour is used to check whether the module is run directly or imported to another file.

The global keyword
global keyword is used to modify the variable outside
of the current scope.

enumerate function in Python
The enumerate function adds counter to an iterable
and returns it

```
for i, item in list1:  
    print(i, item)
```

↳ Prints the items of list 1
with index!

list comprehensions

list comprehension is an elegant way to create lists
based on existing lists

```
list 1 = [ 1, 7, 12, 11, 22]
```

```
list 2 = [i for item in list1 if item > 8]
```


Chapter 10 - Object Oriented Programming

Solving a problem by creating objects is one of the most popular approaches in programming. This is called Object oriented programming.

This concept focuses on using reusable code.

↳ Implements DRY principle

Class

A class is a blueprint for creating objects.

Contains info to
create a valid
Application

Blank
Form

⇒ Filled by an student

Application
of the
Student

class

⇒ object instantiation

Object

Contains info to
create a valid
object

The syntax of a class looks like this :

Class Employee:
methods & variables

[Classname is written in Pascalcase]

Object

An Object is an instantiation of a class. When class is defined, a template (info) is defined. Memory is allocated only after object instantiation.

Objects of a given class can invoke the methods available to it without revealing the implementation details to the user.

→ Abstraction & Encapsulation!

Modelling a problem in OOPs

We identify the following in our problem

Noun → Class → Employee
Adjective → Attributes → name, age, salary
Verbs → Methods → getSalary(), increment()

Class Attributes

An attribute that belongs to the class rather than a particular object.

Example :

Class Employee :

company = "Google" → [specific to each class]

harry = Employee()

→ object instantiation

harry.company

Employee.company = "YouTube" → changing class attribute

Instance Attributes

An attribute that belongs to the Instance (object)

Assuming the class from the previous example :

harry.name = "Harry"

harry.salary = "30K"

⇒ Adding instance attributes

Note : Instance attributes take preference over class attributes during assignment & retrieval

harry.attribute1 → ① Is attribute1 present in object?
② Is attribute1 present in class?

'self' parameter

self refers to the instance of the class
It is automatically passed with a function call from an object

harry.getSalary() → here self is harry
→ equivalent to Employee.getSalary(harry)

The function getsalary is defined as:

```
class Employee:
    company = "Google"
    def getSalary(self):
        print("Salary is not there")
```

Static method

Sometimes we need a function that doesn't use the self parameter. We can define a static method like this:

```
@staticmethod
def greet():
    print("Hello user")
```

→ decorator to mark greet as a static method

--init--() Constructor

--init--() is a special method which is first run as soon as the object is created.

--init--() method is also known as constructor

It takes self argument and can also take further arguments

For Example:

```
class Employee:  
    def __init__(self, name):  
        self.name = name
```

```
    def getSalary(self):  
        ...
```

```
harry = Employee("Harry")
```

→ Object can be instantiated
using constructor like this!