Exception Handling

Date	17 November 2022
Team ID	PNT2022TMID15890
Project Name	Project – Web Phishing Detection

Exception

Exceptions are events that normally occurs during the programming which halts the program in a particular spot and does not go any further. These exceptions are the nightmare for the programmers that causes the program to stop the execution resulting in crash of the application. Luckily, almost every programming languages allows users to handle exception. This process of handling the exceptions is called Exception Handling.

Exception Handling

Exception handling is method handling the unwanted, unforeseen, unexpected events that occurs during the execution of the program. Most programming languages uses 'try-catch', 'try-catch-finally' blocks to handle exception. Python uses the 'try-except', 'try-except-finally' blocks to handle exception.

try – 'try' block is where the code that can cause the exception is placed.

except – 'except' block is where the exception caused is handled.

finally – 'finally' block is where the clean up process is done. It executes regardless the exception occurs.

raise – 'raise' is used to throw a(n) built-in exception or used defined (Custom Exception).

Custom Exception

User can define their own exception and decide to produce any message they want when the exception is raised. **'raise'** can be used to throw the custom or user-defined exception. Most programming languages provides custom exception by Inheriting the base **'Exception'** class.

Exceptions in the developed application:

The application developed by us request the users to input an URL of the website and get 30 features from that website site. There are times when these URLs don't have all the 30 features. These URL causes an exception in the program, thus, crashing the whole application

Exception Handling in the developed application:

Some of the 30 features of the URL like, the URL's length, are always present. These feature checking does not cause any exception. But some features like, IP address, are not always present in the URL. These checking causes exceptions. These kinds of exceptions are handled. When these exceptions are thrown the features checking recognizes them to be a phishing site.

Example of a feature with exception handled:

```
def having_ip(url):
    try:
        domain = urlparse(url).netloc
        ip = socket.gethostbyname(domain)
        if domain == ip:
            return 1
        else:
            return -1
    except socket.gaierror:
        return 0
    except:
        return 1
```