Programming for AI

Abdul Haseeb

BS(AI)-IV

Exception Handling

try: Something that might cause an exception

except: Do this if there **was** an exception

else: Do this if there were **no** exceptions

finally: Do this no matter what happens

Trying to open a file which doesn’t exist

Open the file in try block

Display the exception in except

```python
try:
    file = open("a_file.txt")
except:
    file = open("a_file.txt", "w")
    file.write("Something")
```

We don’t want to fail no matter what..

File was there, but dictionary key didn’t exist

```
#FileNotFound

try:
    file = open("a_file.txt")
    a_dictionary = {"key": "value"}
    print(a_dictionary["sdfsdf"])
except FileNotFoundError:
    file = open("a_file.txt", "w")
    file.write("Something")
```

```
    print(a_dictionary["sdfsdf"])
KeyError: 'sdfsdf'
```

Create Multiple Exceptions

```python
try:
    file = open("a_file.txt")
    a_dictionary = {"key": "value"}
    print(a_dictionary["sdfsdf"])
except FileNotFoundError:
    file = open("a_file.txt", "w")
    file.write("Something")
except KeyError as error_message:
    print(f"The key {error_message} does not exist.")
```

```
The key 'sdfsdf' does not exist.

Process finished with exit code 0
```

```python
try:
    file = open("a_file.txt")
    a_dictionary = {"key": "value"}
    print(a_dictionary["key"])
except FileNotFoundError:
    file = open("a_file.txt", "w")
    file.write("Something")
except KeyError as error_message:
    print(f"The key {error_message} does not exist.")
else:
    content = file.read()
    print(content)
```

```python
try:
    file = open("a_file.txt")
    a_dictionary = {"key": "value"}
    print(a_dictionary["key"])
except FileNotFoundError:
    file = open("a_file.txt", "w")
    file.write("Something")
except KeyError as error_message:
    print(f"The key {error_message} does not exist.")
else:
    content = file.read()
    print(content)
finally:
    file.close()
    print("File was closed.")
```

```python
height = float(input("Height: "))
weight = int(input("Weight: "))

if height > 3:
    raise ValueError("Human Height should not be over 3 meters.")

bmi = weight / height ** 2
print(bmi)
```

Generate Your own exception with raise

```
facebook_posts = [
    {'Likes': 21, 'Comments': 2},
    {'Likes': 13, 'Comments': 2, 'Shares': 1},
    {'Likes': 33, 'Comments': 8, 'Shares': 3},
    {'Comments': 4, 'Shares': 2},
    {'Comments': 1, 'Shares': 1},
    {'Likes': 19, 'Comments': 3}
]
```

- Sum up the likes of facebook post
- Get Ready for KeyError, as few posts don’t have like attribute, use pass keyword

Coding Task

```
try:
    sum=0
    for post in facebook_posts:
            sum=sum+post[‘Likes’]
except KeyError as error:
        pass
```

What is JSON

JSON (JavaScript Object Notation) is a lightweight data interchange format that is easy to read and write for humans and simple to parse and generate for machines.

It is widely used for transmitting data between a server and a web application.

Key Features of JSON

01

Lightweight: Uses a simple text-based structure.

02

Readable: Easily understood by humans.

03

Language-Independent: Supported by many programming languages.

04

Structured Data: Uses key-value pairs (like a dictionary in Python).

JSON Example


```
{
  "name": "John Doe",
   "age": 30,
   "email": "john@example.com",
   "isStudent": false,
    "courses": ["Math", "Science", "History"],
    "address": {
     "street": "123 Main St",
      "city": "New York",
      "zipcode": "10001"
     }
}
```

Importing the JSON Module

```
import json
```

Parsing JSON (Reading JSON Data)

- If you have a JSON string and want to convert it into a Python dictionary, use json.loads():

- json_string = '{"name": "Alice", "age": 25, "city": "New York"}
- data = json.loads(json_string)
- print(data["name"])
- # Output: Alice

Writing JSON (Converting Python to JSON)

- To convert a Python dictionary to a JSON string, use json.dumps():

- python_dict = {"name": "Bob", "age": 30, "city": "Los Angeles"}
- json_output = json.dumps(python_dict)
- print(json_output)

Formatting json Output

- `json_pretty = json.dumps(python_dict, indent=4)`
- `print(json_pretty)`

Reading from a json file

- with open("data.json", "r") as file:
  - data = json.load(file) # Parses JSON into a Python dictionary
  - print(data)

Writing to a json file

- with open("output.json", "w") as file:
  - json.dump(python_dict, file, indent=4)

Modifying json data

- data["age"] = 26 # Updating a value
- print(json.dumps(data, indent=4))

Handling JSON Exceptions

```
try:
    data = json.loads('{"name": "Charlie", "age":}')  # Invalid JSON
except json.JSONDecodeError as e:
    print("JSON decoding failed:", e)
```