Python Files & Exceptions

Besant Technologies

Purpose:

- For taking inputs from other sources.
- For using some parameters from other sources.
- For storing the result into disk.
- For reuse of something later.

Opening & Closing a file:

- open(<path to file>, <mode>)
- Modes:
  - ‘r’ - read mode
  - ‘w’ - write mode
  - ‘a’ - append mode
  - ‘r+’ - reading and writing
  - ‘w+’ - writing and reading
  - ‘Wb’, ‘rb’ - for reading and writing binary files

* with statement

* close()

Reading & Writing:

- f.read()
- f.read(<number of characters>)
- f.readline()
- f.readlines()
- for line in f:
- f.write()
- f.writelines()

Special functions:

- seek()
- tell()

JSON files:

- JSON - Javascript object notation
- Lightweight format.
- Independent of platform.
- Faster to process.
- “import json”
- Common storage format.

Python to Json and vice versa

- Json to python: json.loads()
- Python to json: json.dumps()

Python	JSON
dict	Object
list	Array
tuple	Array
str	String
int	Number
float	Number
True	true
False	false
None	null

JSON Files:

- .json extension.
- json.dump(<python_variable>, file_object)
- <python_variable> = json.load(file_object)

Errors:

- Types:
  - Syntax errors
  - Run-time errors
  - Linker error
  - Logical error

Syntax errors:

- Violate the rules of programming
- Ex: Indentation, missing colon.
- Detected by interpreter

Run-time errors

- Occur during program execution.
- Ex: Divisible by zero error.
- Can’t be detected by Interpreter.

Linker errors:

- When some dependencies can’t be linked.
- Not able to create an executable file.
- Ex: Error in importing
- Ex: Wrong function prototype

Logical errors:

- Error in logic.
- Output is not as expected.
- Ex: Infinite loop, missing increment in loops etc.,

try-except:

- Exception handling
- try: Keep the error proning code
- except: keep the statement to handle in case of error.
- finally: This block will get executed irrespective of exception.