* File Handling and Error handling :-

File handling :-

File handling in python allows us to create, read, write and delete files. it is used to store data permanently on the disk, so the data remains even after the program ends.

Built-in function

syntax :-

```
file = open ("filename", "mode").
```

Ex :-

```
f = open ("data.txt", "w").
```

we use the open() function to work with files.

Common File Modes :-

Mode	Meaning	Description
"r"	Read	Opens files for reading (error if not found).
"w"	write	opens files for writing (erases existing data)
"a"	Append	open file to add new data at the end.
"b"	Binary	used for binary files (images, vedios, etc).
"t"	Text	Default mode - for text files
"rb"	Read Binary	opens binary file for reading.
"wb"	write Binary	opens binary file for writing.
"wt"	write Text	opens text file for writing.

File paths :-

1. **Relative path :-**

Refers to the file location relative to the current working directory.

ex:-
```
open ("data.txt", "r")
```

2. **Absolute path :-**

Refers to the complete location of the file in your system.

ex:-
```
open ("sample.txt", "r")
```

ex:-

* ```
d = open (r "c:\users\one drive\desktop\sample.rtf", "r")
d.read()
d.close()
```

* ```
d1 = open ("demo.txt", "w")
t = "good morning, good afternoon, good evening"
d1.write(t)
d1.close()
```

* os.remove (r "c:\users\bnage\desktop\my_data.text", "")
os

Error Handling :-

Errors are issues that stop a program from running properly. Error handling means detecting and managing those errors gracefully - so that the program doesn't crash unexpectedly.

Common Error Types :-

	Error Type	Description	example.
1.	Syntax Error	occurs when python code syntax is incorrect	print ("Hello" x (missing bracket)
2.	Name Error	using a variable that doesn't exist	print (x) x (x not defined)
3.	indentation error	incorrect indentation (spaces / tabs)	if True : \n print ("Hi") x
4.	zero division error	dividing a num by zero	a = 10 / 0 x
5.	Type Error	using incompatible data type	5 + "hello" x
6.	Value Error	wrong value passed to a function	int ("abc") x
7.	Index Error	Accessing a list element out of range	num = [1,2,3]; print (num [5]) x
8.	Key Error	Accessing a non-existing key in a dictionary	data = {"name": "A"}; print (data ["age"]) x
9.	File Not Found Error	Trying to open a file that doesn't exist	open ("abc.txt") x

1. Syntax error:-

when you write code that violates python syntax values.

Ex:- # Missing closing parenthesis.

```
print ("Hello "      x syntax error
```

correct:- print ("Hello")

2. Name error

using a variable or function name that is not defined.

Ex:-

```
print (x)      # x name error : name 'x' is not defined.
```

correct:-

```
x =10
print(x)
```

3. indentation error

incorrect indentation (python use indentation to define blocks).

Ex:-

```
if True:
print ("Hello")      # indentation error.
```

correct:-

```
if true:
    print ("Hello")
```

4. Zero Division Error

Dividing a number by zero

Ex:- a = 10 /0 # zero Division error.

correct:-

```
a = 10/2
print (a)
```

5. Type error:-

performing an operation on mismatched data types:

Ex:-

```
a = 5 + "Hello"      # Type error
```

correct:

```
a = 5 +10
print (a)
```

6. Value Error

When a function receives the right type but wrong value.

ex:-
```
num = int ("abc")        # Value Error : invalid literal for int()
```

correct:-
```
num = int ("123")
print (num)
```

7. Index Error

Accessing an index that doesn't exist in a list.

ex:-
```
num = [1,2,3]
print (num [5])          # Index error : list index out of range.
```

correct:-
```
num = [1,2,3]
print (num [2])          # O/P :- 3.
```

8. Key Error

Accessing a non-existent key in a dictionary

ex:-
```
stu = {"Name" : "Nagi", "age" : 20}
print (stu ["grade"])    # Key error : 'grade'
```

correct:-
```
print (stu ["Name"])
```

9. File Not-found Error

Trying to open a file that doesn't exist

ex:-
```
file = open ('abc.txt', 'r')    # File Not-found error
```

correct:-
```
File = open ("data.txt", "r")   # make sure the file exists.
print (file.read())
File.close()
```