Exception handling in python

Def Exception :

Exception is unwanted error that are occure due to the mistake of developers. It always comes when your program is running.

- Error in python can be of two types that is Syntax error and Exceptions.
- Error are problems in program due to which the program will stop the execution.
- To handle the exception python uses the keywords like try and except block.

What are the different types of exceptions in python:

In python, there are servel built in python exceptions that can be raised when error occurs during the execution of a program.

1) SyntaxError:

- This exception is raised when the interpreter encounters a syntax error in the code.
- Such as a misspelled keyword, missing colon etc...

2) TypeError:

- Raised when operation or function is applied to an object of an inappropriate type.

Example:

```
print("abc" + 136)
```

3) ValueError:

- Raised when a function receives an argument of the correct type but inappropriate value.

Example:

```
print(int("abc"))
```

4) IndexError:

- Raised when trying to access an index tha is out of range.

Example:

```
lst = [1,2,3,4,5]
print(lst[7])
```

5) KeyError:

- Raised when trying to access a key that does not exist in a dictionary.

Example:

```
d = {"city":"Pune"}
print(d["name"])
```

6) AttributeError:

- Raised when an object does not have the specified attribute.

Example:

```
    lst = [1,2,3,4,5]
    lst.appendd(100) # here i am writing wrong method
    print(lst)
```

7) FileNotFoundError:

- Raised when a file operations because the file does not exist.

Example:

```
    open(r"D:\test1.txt","r")
```

In above statement i am looking for the file test1.txt on the D drive, but this type of file does not exists on that drive, so it will generate the exception FileNotFoundError: [Errno 2] No such file or directory: 'D:\\test1.txt'.

8) IndentationError:

Raised when the code is not properly aligned.

Example:

```
def printData():
print("Welcome")
```

8) IOError(input/output error):

- This exception is raised when an I/O operation, such as reading or writing a file fails due to an input/output error.

9) ImportError/ ModuleNotFoundError:

- Raised when an import statement fails.

Example:

```
import arrays ---> arrays module not present. The correct module name is array
```

10) ZeroDivisionError:

```
-----------------------
    - This exception is raised when an attempt is made to divide a number by zero.

*Q. What is the differenece between Syntax Error and Exception*
-----------------------------------------------------------------------
- Syntax Error:
--------------------
1. As the name suggest this error is caused by the wrong syntax in the code. It leads to the termination of the program.
Example:
-----------
amount = 200
if amount > 300
    print("You can purchase the product")
else
    print("Check money")

In above code, there is syntax error in the code. The if statement should be followed by a colon(:).

Exceptions:
----------------
- Exceptions are raised when the program is syntactically correct, but the code results in an error. This error does not
  stop the execution of the program, however it changes the normal flow of the program.
Example:
-------------
marks = 100
res = marks / 0
print(res)

add = marks + 10
print(add)

* How you can avoid the exceptions:*
---------------------------------------------
- Try and except statements are used to catch and handle exceptions in python.
- The try clause contains the code that can raise the exception, while the except clause contains the code lines that handle
  the exception.
- You can use only one try statement but you can use multiple except blocks.
- The try block contains atleast one except block. You dont write try without except.

Example:
------------------
marks = 100
try:
    res = marks / 0
    print(res)
except ZeroDivisionError as err :
    print(err)
finally:
    print("hi welcome")

add = marks + 10
print(add)

-------------------------------------------------------------------------------------------
*Try with else block*
--------------------
- The else clause in Python exception handling is used to specify a block of code that should run only if no exceptions were raised.

Syntax:
----------
try:
    #Code that may raise the exception
except:
    #Code to handle the exception
else:
    #Code that runs if no exception was raised.

Example:
-----------
def divide_numbers(num1,num2):
    try:
        result = num1 / num2
    except ZeroDivisionError:
        print("Error: Division by zero is not allowed")
    else:
        print("Division successful..! The result is = ", result)

divide_numbers(10,5)
divide_numbers(5,0)

-----------------------------------------------------------------------------------------------------------------------
Raising Exception:
--------------------
- The raise statement allows the programmer to force a specific exception to occur.
- In Python you can raise an exception using the raise.
- The following is the basic syntax of built-in exception
    raise ExceptionType("Error message")
```

- You can also define your own exceptions by subclassing the **Exception** class:
- The following **is** the syntax of custom_exception

```
    class CustomeError(Exception):
        def __init__(self, message):
            super().__init__(message)

    #Raise the custom exception
    raise CustomError("This is a custom error")
```

```
Example -->01 ---> builtIn
------------------------------
def divide_numbers(num1, num2):
    if num2 ==0:
        raise ZeroDivisionError("You cant divide by zero")
    return num1 / num2

try:
    result = divide_numbers(10,5)
    print("The result is = ",result)
except ZeroDivisionError as err:
    print("Error:",err)
else:
    print("Finally i am executed")


Example ---> 02 ---> Custom Exception
--------------------------------------
class NegativeNumberError(Exception):
    def __init__(self, message="Negative numbers are not allowed"):
        super().__init__(message)


def square_root(number):
    if number < 0:
        raise NegativeNumberError("Cannot compute the square root of negative number")
    return number ** 0.5
try:
    result = square_root(-25)
    print("The result is = ", result)
except NegativeNumberError as err:
    print("Error:", err)

================================================================================================
```