1- Exceptions Handling in Python

Errors and exceptions are common terms in Python programming. An error is a problem that prevents the program from running, such as a syntax error. An exception is a problem that occurs during the execution of the program, such as a division by zero. You can handle exceptions using the try and except block, which allows you to catch and recover from errors gracefully. You can also raise your own exceptions using the raise keyword, which allows you to signal an error condition to the caller of your function

1.1- Handling exceptions

In [1]:
```
1/0
```

```
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
C:\Users\EUROTE~1\AppData\Local\Temp/ipykernel_7512/2354412189.py in <module>
----> 1 1/0

ZeroDivisionError: division by zero
```

In [2]:
```
# Catching errors with try and except
try:
    1/0
except ZeroDivisionError:
    print("You cannot divide by zero!")
```

```
You cannot divide by zero!
```

In [3]:
```
# Bare excepts
try:
    1/0
except:
    print("You cannot divide by zero!")
```

```
You cannot divide by zero!
```

In [4]:
```python
# Dictionary example
my_dict = {"a":1, "b":2, "c":3}
try:
    value = my_dict["d"]
except KeyError:
    print("That key does not exist!")
```

```
That key does not exist!
```

In [5]:
```python
# List error message
my_list = [1, 2, 3, 4, 5,6]
try:
    my_list[6]
except IndexError:
    print("That index is not in the list!")
```

```
That index is not in the list!
```

In [6]:
```python
# Catching multiple exceptions
my_dict = {"a":1, "b":2, "c":3}
try:
    value = my_dict["d"]
except IndexError:
    print("This index does not exist!")
except KeyError:
    print("This key is not in the dictionary!")
except:
    print("Some other error occurred!")
```

```
This key is not in the dictionary!
```

In [7]:
```python
#Can we use () or Parenthesis
try:
    value = my_dict["d"]
except(IndexError, KeyError):
    print ("An IndexError or KeyError occurred!")  # this method is not recommended  the method above is good
```

```
An IndexError or KeyError occurred!
```

In [8]:
```
#A statement “finally” to handle exceptions
# example for finally

my_dict = {"a":1, "b":2, "c":3}
try:
    value = my_dict["d"]
except KeyError:
    print("A KeyError occurred!")
finally:
    print("The finally statement has executed!")
```

```
A KeyError occurred!
The finally statement has executed!
```

In [9]:
```
# try, except, or else!The try/except statement also has an else clause. The else will only run if there are n
# Example

dict = {"a":1, "b":2, "c":3}
try:
    value = dict["a"]
except KeyError:
    print("A KeyError occured!")
else:
    print("No error occured!")
```

```
No error occured!
```

In [10]:
```
dict = {"a":1, "b":2, "c":3}
try:
    value = dict["a"]
except KeyError:
    print("A KeyError occured!")
else:
    print("No error occured!")
finally:
    print("The finally statement ran!")
```

```
No error occured!
The finally statement ran!
```

2- different common exceptions occurred in python

2.1-An AttributeError

An AttributeErro in Python is raised when you try to call an attribute of an object whose type does not support that method. For example, if you try to use the append() method on an integer, you will get an AttributeError because integers do not support append().

In [11]:
```
# Example
i = 1
i.append(2)
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
C:\Users\EUROTE~1\AppData\Local\Temp/ipykernel_7512/3304352496.py in <module>
      1 # Example
      2 i = 1
----> 3 i.append(2)

AttributeError: 'int' object has no attribute 'append'
```

2.2- IOError

Raised when an I/O operation (such as reading or writing a file) fails.

In [12]:
```
# Example
try:
    with open("non_existent_file.txt", "r") as f:
        content = f.read()
except IOError:
    print("Error:")
```

```
Error:
```

2.3- ImportError

Raised when an imported module cannot be found or loaded.

In [13]:
```
try:
    import non_existent_module
except ImportError:
    print("Error:")
```

```
Error:
```

2.4- IndexError

Raised when an index for a sequence (list, tuple, etc.) is out of range.

In [14]:
```
try:
    my_list = [1, 2, 3]
    print(my_list[10])
except IndexError :
    print("Error:")
```

```
Error:
```

2.5- KeyError

Raised when a dictionary key is not found.

In [15]:
```
try:
    my_dict = {"a": 1, "b": 2}
    value = my_dict["c"]
except KeyError:
    print("Error:")
```

```
Error:
```

2.6- KeyboardInterrupt

Raised when the user interrupts the program's execution (e.g., by pressing Ctrl+C).

In [ ]:
```
try:
    while True:
        pass
except KeyboardInterrupt:
    print("Execution interrupted by user.")
```

2.7- NameError

Raised when a local or global name is not found.

In [16]:
```
try:
    print(non_existent_variable)
except NameError:
    print("Error:")
```

```
Error:
```

2.8- OSError

Raised for operating system-related errors.

In [17]:
```
try:
    with open("/non_existent_file.txt", "r") as f:
        content = f.read()
except OSError:
    print("Error:")
```

```
Error:
```

2.9- SyntaxError

raised when there's a syntax error in the code.

In [18]:
```
try:
    eval("print 'Hello, world!'")
except SyntaxError:
    print("Error:")
```

```
Error:
```

2.10- TypeError

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

In [19]:
```
try:
    result = "hello" + 42
except TypeError:
    print("Error:")
```

```
Error:
```

2.11- ValueError

Raised when a function receives an argument of correct type but with an invalid value.

In [20]:
```
try:
    num = int("abc")
except ValueError:
    print("Error:")
```

```
Error:
```

2.12- ZeroDivisionError

Raised when division or modulo by zero is attempted.

In [21]:
```
try:
    result = 10 / 0
except ZeroDivisionError:
    print("Error:")
```

```
Error:
```

In [ ]:
```

```