4_exceptions

November 21, 2023

0.1 Exceptions

In Python errors are managed with a special language construct called "Exceptions". When errors occur exceptions can be raised, which interrupts the normal program flow and fallback to somewhere else in the code where the closest try-except statement is defined.

To generate an exception we can use the `raise` statement, which takes an argument that must be an instance of the class `BaseException` or a class derived from it.

[3]:
```
raise Exception("description of the error")
```

```
---------------------------------------------------------------------------
Exception                                 Traceback (most recent call last)
<ipython-input-3-c32f93e4dfa0> in <module>
----> 1 raise Exception("description of the error")

Exception: description of the error
```

A typical use of exceptions is to abort functions when some error condition occurs, for example:

```
def my_function(arguments):

    if not verify(arguments):
        raise Exception("Invalid arguments")

    # rest of the code goes here
```

To gracefully catch errors that are generated by functions and class methods, or by the Python interpreter itself, use the `try` and `except` statements:

```
try:
    # normal code goes here
except:
    # code for error handling goes here
    # this code is not executed unless the code
    # above generated an error
```

For example:

[4]:
```
try:
    print("test")
    # generate an error: the variable test is not defined
    print(test)
except:
    print("Caught an exception")
```

```
test
Caught an exception
```

To get information about the error, we can access the `Exception` class instance that describes the exception by using for example:

`except Exception as e:`

[5]:
```
try:
    print("test")
    # generate an error: the variable test is not defined
    print(test)
except Exception as e:
    print("Caught an exception:" + str(e))
```

```
test
Caught an exception:name 'test' is not defined
```