PROGRAMMING IN PYTHON I

Unit 04: Exceptions

Michael Widrich
Institute for Machine Learning

JOHANNES KEPLER
UNIVERSITY LINZ

ERROR HANDLING AND EXCEPTIONS

Motivation

- In programming, we sometimes encounter problems that would crash our program
  - Wrong datatype used as input by user
  - Use-case we did not consider
  - Syntax- or other errors in our code
- The severity of such a problem depends on how well the program can handle the error
- Proper error handling can:
  - Give the user clear information on what went wrong
  - Terminate the program in a proper way (e.g. closing all open files, writing a logfile, saving trained ML models, . . .)
  - Fix the error and continue with the program execution (not always desired!)

Exceptions in Python

- In Python, errors raise exceptions
  - If an error occurs, an exception is created (“raised”)
  - An exception carries information on what went wrong
  - There are different exception types (we can also create our own exception types)
- Exceptions can be caught and dealt with in the program
- If an exception is raised, the program execution will jump to where the exception is caught or to the end of the program
  - In Python, exceptions have a notion of control-flow tools, such as if-else code blocks
- We can raise exceptions ourselves

Exceptions in Python: Syntax

- We can raise an exception with the raise statement:
  - This raises a ValueError exception:
    ```
    raise ValueError("Some error message")
    ```
- To catch an exception, we have to be prepared:
  - We have to use a try code block in which we can catch the exception…
  - …followed by an except code block, in which we specify our exception handling
  - We can also follow it with a finally code block, to unconditionally execute code (e.g. for closing/saving files)

Exceptions in Python: Example 1

- Here we catch an exception, print a warning, and continue with our program

```
try:
    a = 1 + 'f'  # This will raise a "TypeError"
    a += 2  # This will not be executed
except TypeError as ex:
    # We will land here if TypeError was raised
    print(f"We caught the exception {ex}!")
    a = 1 + 2
a *= 2  # This will be executed
```

Exceptions in Python: Example 2

- Here we catch an exception, print a warning, and raise the exception again to terminate our program

```
try:
    a = 1 + 'f'  # This will raise a "TypeError"
    a += 2  # This will not be executed
except TypeError as ex:
    # We will land here if TypeError was raised
    print(f"We caught the exception {ex}!")
    # We could e.g. close/save files here
    raise ex
a *= 2  # This will not be executed
```