Cheat sheet: Python exceptions & modules

Xavier Clerc – xavier.clerc@enpc.fr – October 2020

Exceptions

Declaration

```
class MyException(Exception):
  def __init__(self, ...):
    super().__init__()
    # ...
  def __repr__(self): # commonly defined
    # ...
  def __str__(self): # commonly defined
    # ...
  # ...
```

Raise

```
raise MyException(...)
raise MyException # () elided
raise # only in a handling block,
      # re-raises the caught exception
```

Basic handling

```
try:
  # ...
  # protected block
  # ...
except: # catches any exception
  # ...
  # handling block
  # ...

try:
  # ...
  # protected block
  # ...
except E1: # catches instances of E1
  # ...
  # handling block
  # ...
except E2: # catches instances of E2
  # ...
  # handling block
  # ...
except: # catches any exception
  # ...
  # handling block
  # ...
```

```
try:
  # ...
  # protected block
  # ...
except E as e: # e references the instance
  # ...
  # handling block
  # ...
```

Advanced handling

```
try:
  # ...
except ...:
  # ...
else:
  # ...
  # executed after the try block if no exception is
  # raised during its execution
  # ...

try:
  # ...
except ...:
  # ...
finally:
  # ...
  # executed after either the try or the except
  # block has been successfully executed
  # ...
```

If both `else` and `finally` are present, `else` is executed first.

Built-in exceptions

Non-exhaustive list:

- `AttributeError`: when an attribute reference is invalid
- `FileExistsError`: when a file already exists
- `FileNotFoundError`: when a file does not exist
- `ImportError`: when an import fails
- `IndexError`: when an index is out of bounds
- `KeyError`: when a key does not exist
- `ModuleNotFoundError`: when a module does not exist
- `NameError`: when a name is not bound
- `RecursionError`: when the max recursion depth is reached
- `ValueError`: when a function/method receives an invalid value
- `ZeroDivisionError`: when dividing by zero (integers)

Resources

```
with expr0 as id0, ..., exprn as idn:
  # ...

# or equivalently:

with expr0 as id0:
  # ...
  with exprn as idn:
    # ...
```

The resources created by the `expr`*i* are automatically closed / cleaned up when the program exits the `with` block, even if the block is exited due to an exception.

```
with open("my-file-name.ext", ...) as f:
  # ...
  # f can be used to read/write from/to the file
  # ...
# the file is automatically closed when the with
# block ends
```

Modules

Declaration

A module is simply a `.py` file. The module hierarchy is based on the file hierarchy, *i.e.* the file named `a/b.py` defines the `a.b` module. The search path is `sys.path` (includes `.` and the `PYTHONPATH` environment variable).

Qualified use (preferred)

```
import module
module.element

import module as m
m.element

import module1, module2 as m, ...
module1.element
m.element
```

Unqualified use (discouraged)

```
from module import element
element

from module import element as e
e

from module import *
element
other_element
```