



PyQB

Monga

Iterators and  
generators

Exception  
handling

# Programming in Python<sup>1</sup>

Mattia Monga

Dip. di Informatica  
Università degli Studi di Milano, Italia  
mattia.monga@unimi.it

Academic year 2025/26, I semester

<sup>1</sup>© 2025 M. Monga. Creative Commons Attribuzione — Condividi allo stesso modo 4.0 Internazionale. <http://creativecommons.org/licenses/by-sa/4.0/deed.it>



PyQB

Monga

Iterators and  
generators

Exception  
handling

## Lecture XIX: More pandas



PyQB

Monga

Iterators and  
generators

Exception  
handling

## Iterators

Object can be iterable. Python defines the iterator protocol as:

- `iterator.__iter__()` Return the iterator object itself. This is required to allow both containers and iterators to be used with the `for` and `in` statements.
- `iterator.__next__()` Return the next item from the container. If there are no further items, raise the `StopIteration` exception.



PyQB

Monga

Iterators and  
generators

Exception  
handling

## Notable iterators

Built-in lists, tuples, ranges, sets, dicts are iterators.

- Numpy arrays
- Pandas Series and DataFrames

## Generators



PyQB

Monga

Iterators and  
generators

Exception  
handling

```
from collections.abc import Generator

def mygenerator() -> Generator[int]:
    for i in [1, 6, 70, 2]:
        yield i
    print('Ended')  # Just to see when it reaches this
                    ↪ point

g = mygenerator()

print(g)  # not useful
print(next(g))
print(next(g))
print(next(g))
print(next(g))
print(next(g))  # Exception
```

142

## Pandas DataFrame



PyQB

Monga

Iterators and  
generators

Exception  
handling

Be careful: the default iteration is on **column names** (similar to dicts, which iterate on keys).

- `iterrows()`: Iterate over the rows of a DataFrame as (index, Series) pairs. This converts the rows to Series objects, which can change the dtypes and has some performance implications.
- `itertuples()`: Iterate over the rows of a DataFrame as namedtuples of the values. This is a lot faster than `iterrows()`, and is in most cases preferable to use to iterate over the values of a DataFrame.

Iterating is **slow**: whenever possible try to use vectorized operation or **function application**.

143

## Pandas function application



PyQB

Monga

Iterators and  
generators

Exception  
handling

```
# apply the function to each column
df.apply(lambda col: col.mean() + 3)

# apply the function to each row
df.apply(lambda row: row + 3, axis=1)
```

144

## Pandas query



PyQB

Monga

Iterators and  
generators

Exception  
handling

```
df[df['A A'] > 3]
```

# equivalent to this (backticks because of the space)  
`df.query('`A A` > 3')`

# query can also refer to the index  
`df.query('index >= 15')`

# same as  
`df[15:]`

145



PyQB

Monga

Iterators and  
generatorsException  
handling

## Lecture XX: Exception handling

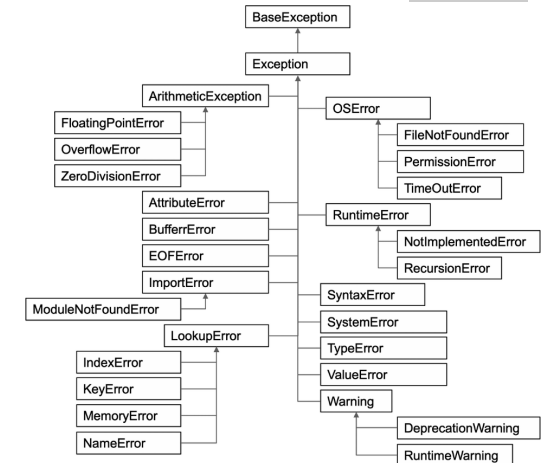
146

## Exceptions



PyQB

- Exceptions and Errors are object **raised** (or thrown) in the middle of an anomalous computation.
- Exceptions change the control flow: the control passes to the “closer” handler, if it exists: otherwise it **aborts**.



147



PyQB

Monga

Iterators and  
generatorsException  
handling

## Exception handling

Exceptions can be handled: the strategy is normally an “organized panic” in which the programmer tidies up the environment and exits.

```

danger()
# An exception in danger
# aborts the program

try:
    danger()
except:
    # An exception in danger
    # it's handled here

try:
    danger()
except OverflowError as e:
    # An exception in danger
    # it's handled here
    # The object is referred by
    ↪ e
finally:
    # This is executed in any
    ↪ case
  
```

148

## Raising an exception



PyQB

Monga

Iterators and  
generatorsException  
handling

To explicitly raise an exception, use the `raise` statement

```

if something == WRONG:
    raise ValueError(f'The value {something} is wrong!')
  
```

Assertions are a disciplined way to raise exceptions.

149