Python Programming Language Foundation

Session 7

Session overview

- Exceptions
- Context managers
- Iterators
- Generators

Exceptions

Exceptions

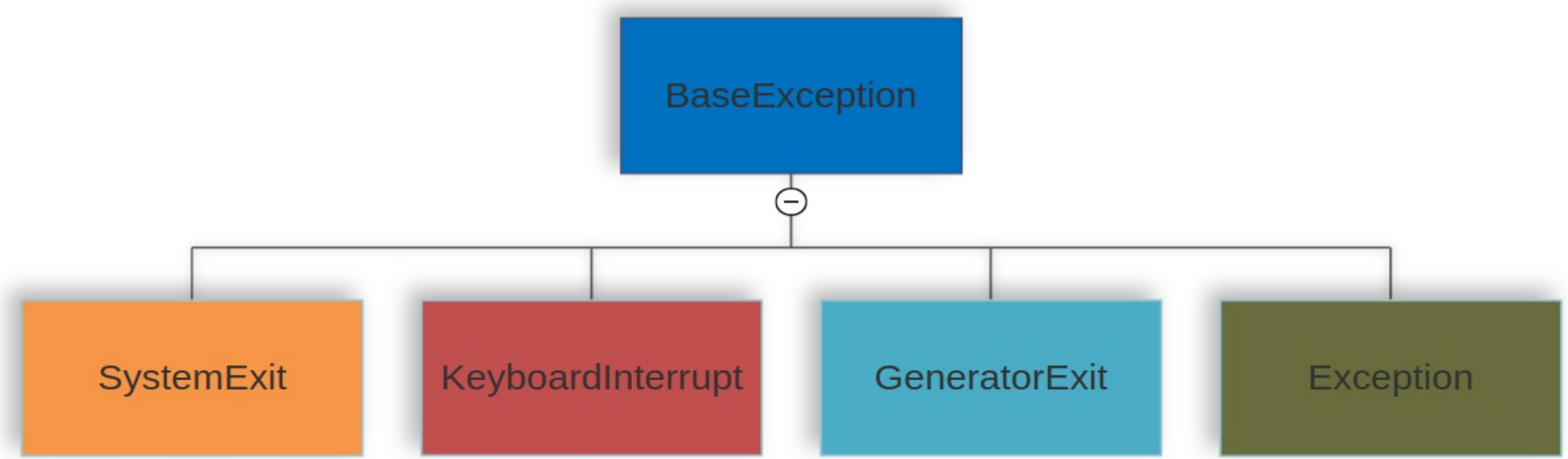
An event,
which occurs during the execution of a program that disrupts the normal flow of the program's instructions

Python object
that represents an error situation

There are (at least) two distinguishable kinds of errors: *syntax errors* and *exceptions*

It is possible to write programs that handle selected exceptions

Standard exception hierarchy



https://docs.python.org/3/library/exceptions.html#exception-hierarchy

Standard exception hierarchy

- StopIteration
- StopAsyncIteration
- ArithmeticError
- AssertionError
- AttributeError
- BufferError
- EOFError
- ImportError
- LookupError
- MemoryError
- NameError
- OSError
- ReferenceError
- RuntimeError
- SyntaxError
- SystemError
- TypeError
- ValueError
- Warning

https://docs.python.org/3/library/exceptions.html#exception-hierarchy

Standard exception hierarchy

- StopIteration
- StopAsyncIteration
- ArithmeticError
- AssertionError
- AttributeError
- BufferError
- EOFError
- ImportError
- LookupError
- MemoryError
- NameError
- OSError
- ReferenceError
- RuntimeError
- SyntaxError
- SystemError
- TypeError
- ValueError
- **Warning**

https://docs.python.org/3/library/exceptions.html#exception-hierarchy

Exception generation

```
raise [exception_type|exception_object]
```

Handling exceptions

```
try:
    ...  # You do your operations here
except Exception1:
    ...  # Exception1 has been raised
except Exception2:
    ...  # Exception2 has been raised
else:
    ...  # No exceptions has been raised
finally:
    ...  # Always executed no matter what
```

Handling multiple exceptions in a same manner

```
try:
    ...  # You do your operations here
except (Exception1[, Exception2[, ...]]) as err:
    ...  # Any exception from the given list has been raised
```

Exception re-raising

```
raise [exception_type|exception_object]
```

```
class CustomException(exception_type)
```

Best practice

except Exception as *exc*

raise Exception *from exc*

Specify types of caught exceptions

Use *finally* and *else* blocks

Iterators

Iterable & Iterator

Iterable

Iterator

An **object** capable of returning its **members one at a time**

https://docs.python.org/3/glossary.html#term-iterable

Iterable examples

list/tuple

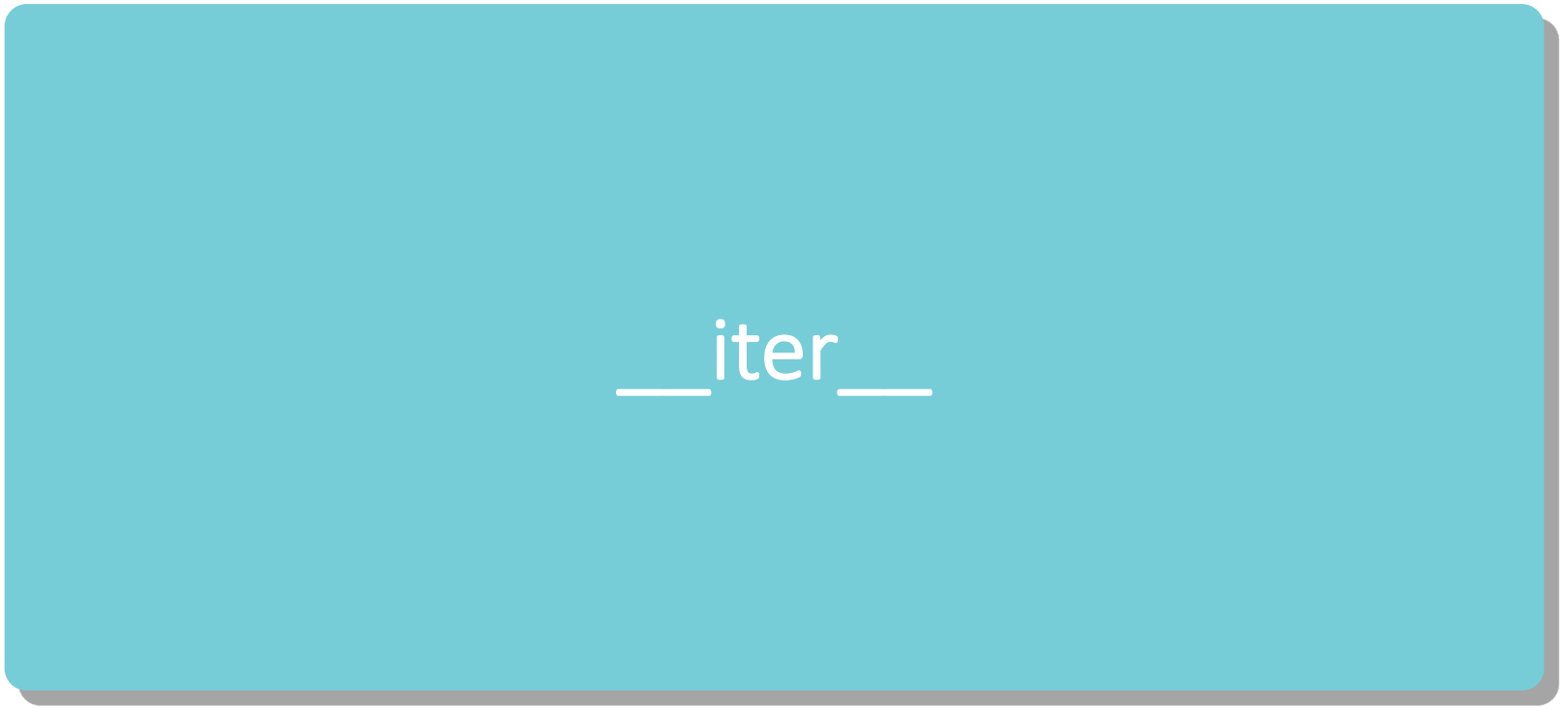
str

tuples

File objects

https://docs.python.org/3/glossary.html#term-iterable

Iterable implementation



https://docs.python.org/3/glossary.html#term-iterable

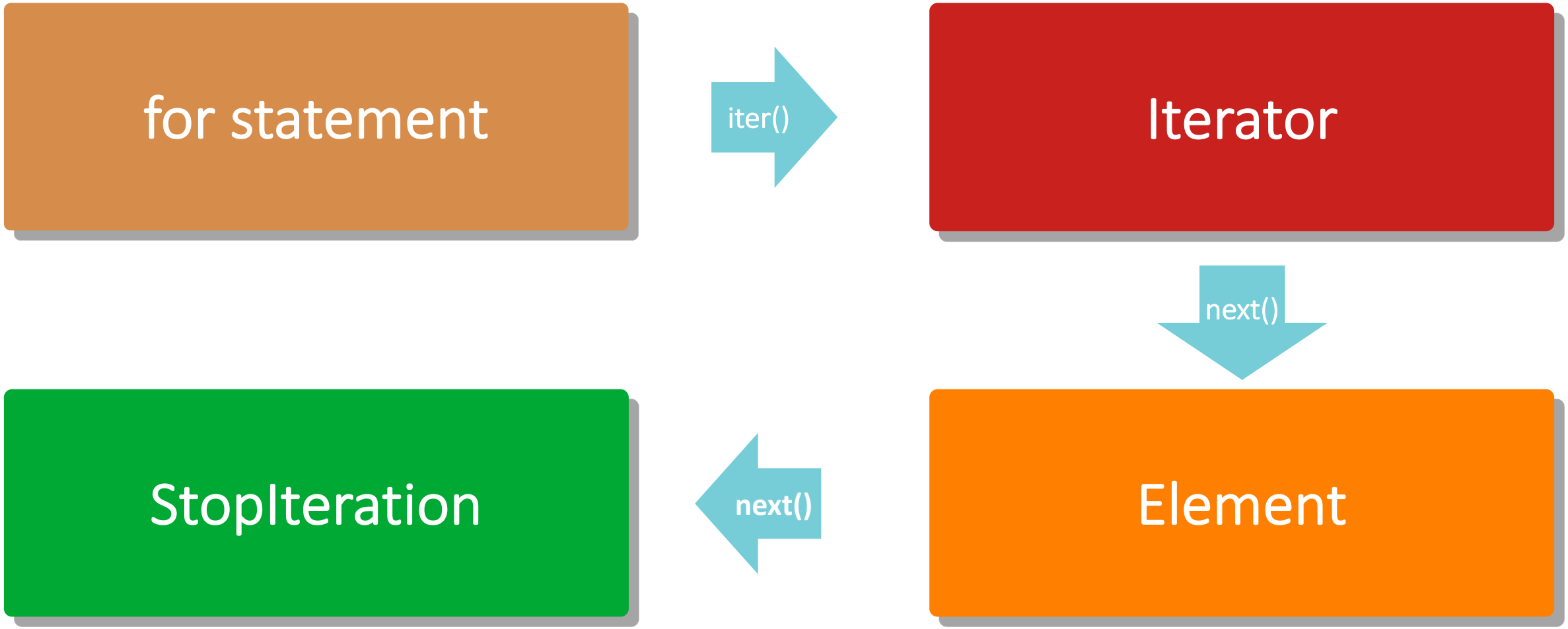
An **object** representing a **stream of data**. **Repeated** calls to the iterator's **__next__()** method return **successive items** in the stream

https://docs.python.org/3/glossary.html#term-iterable

A class that wants to be an **iterator** should implement **two methods**: a **__next__()** method, and an **__iter__()** method that **returns self**

https://docs.python.org/3/glossary.html#term-iterable

How does it work?



https://docs.python.org/3/glossary.html#term-iterable

Generators

Generator is a kind of **iterator** which does not know anything about collection he iterates, **only** how to create next element (e.g. generates)

https://www.python.org/dev/peps/pep-0255/

It **looks** like a **normal function** except that it contains one or more **yield expressions**

https://docs.python.org/3/glossary.html#term-generator

```
def function_name([arguments]):
    [function_expression]
    yield_expression
    [function_expression]
```

Context Managers

A **context manager** is an object that defines the runtime context to be established when executing a `**with**` statement

Context manager

Context manager as a class

Context manager as a function

`with` statement

```
with context [as target]:
    # code block
```

Context manager as class

__enter__

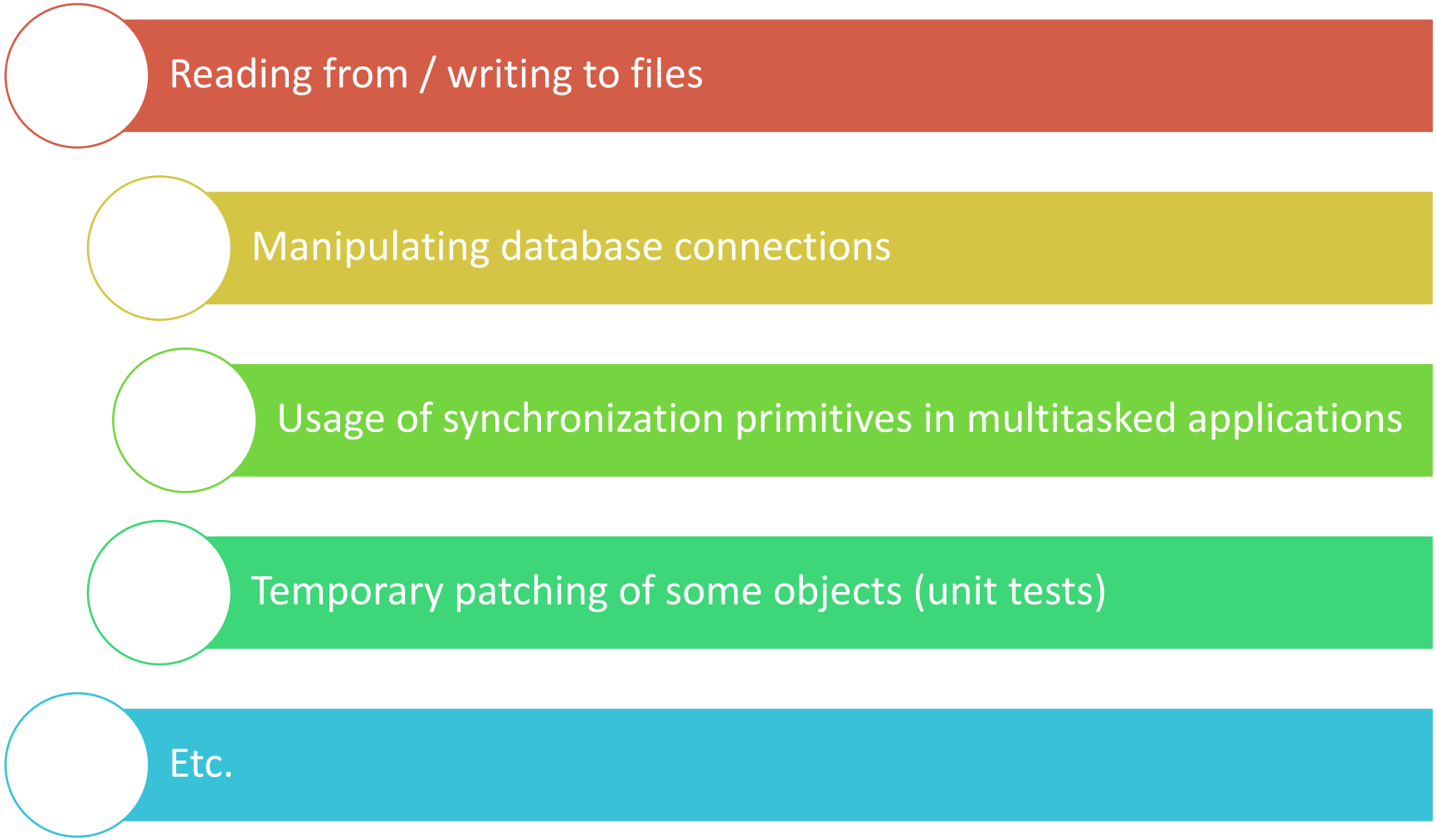
__exit__

```
@contextmanager
def function_name([arguments]):
    [function_expression]
    yield_expression
    [function_expression]
```

contextlib.	
	contextmanager
	asynccontextmanager
	closing
	nullcontext
	suppress
	redirect_stdout
	redirect_stderr

https://docs.python.org/3/library/contextlib.html

Some popular use cases



Thanks for attention

Questions?