SI100B Introduction to Information Science and Technology **Python Programming**

张海鹏 Haipeng Zhang
School of Information Science and Technology
ShanghaiTech University

HW2 and Tutorial

- HW2 to be released Friday, Oct 2
- Dues 23:59:59, Sunday, Oct 18
- Last Tutorial: 8PM Friday, Oct 2, will be recorded
- Office Hour: Oct 9 to Oct 18

Learning Objectives

- Miscellaneous
  - Iterator
  - User-defined exception

Iterator

```
for element in [1, 2, 3]:
    print(element)
for element in (1, 2, 3):
    print(element)
for key in {'one':1, 'two':2}:
    print(key)
```

1. The for statement calls iter() on the container object
2. iter() returns an iterator object that defines __next__() which accesses elements in the container one at a time
3. When there are no more elements, __next__() raises a StopIteration exception which tells for loop to terminate

The iterator protocol

```
>>> s = 'abc'
>>> it = iter(s)
>>> it
<iterator object at 0x00A1DB50>
>>> next(it)
'a'
>>> next(it)
'b'
>>> next(it)
'c'
>>> next(it)
Traceback (most recent call last):
File "<stdin>", line 1, in <module>
next(it)
StopIteration
```

User-defined class supporting iteration

- Override __iter__() method which returns an object with a __next__() method

- If the class defines __next__(), then __iter__() can just return self

User-defined class supporting iteration

```
class Reverse:
    """Iterator for looping over a sequence 
backwards."""
    def __init__(self, data):
        self.data = data
        self.index = len(data)

    def __iter__(self):
        return self

    def __next__(self):
        if self.index == 0:
            raise StopIteration
        self.index = self.index - 1
        return self.data[self.index]
```

User-defined class supporting iteration

```
>>> rev = Reverse('spam')
>>> iter(rev)
<__main__.Reverse object at 0x00A1DB50>
>>> for char in rev:
... print(char)
...
m
a
p
s
```

Iterator creation via map

map(function, iterable,...)

```
>>> x = map(lambda x: x ** 2, range(3))
>>> next(x)
0
>>> next(x)
1
>>> next(x)
4
>>> next(x)
Traceback (most recent call last):
File "<pyshell#31>", line 1, in <module>
next(x)
StopIteration
>>>
```

Iterator creation via Map

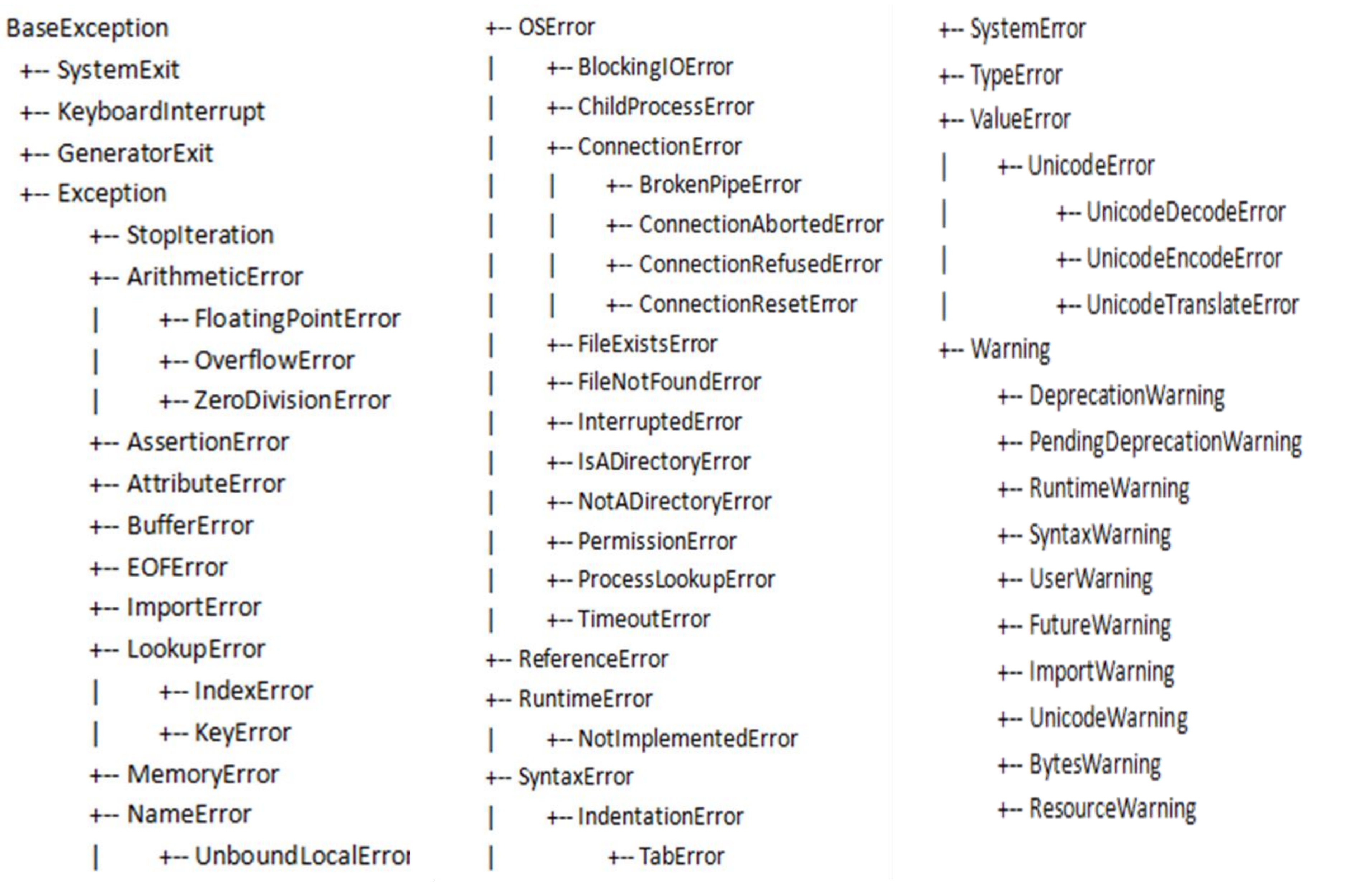
map(function, iterable,...)

```
>>> x = map(lambda x,y: x+y, range(3),range(2,6))
>>> next(x)
2
>>> next(x)
4
>>> next(x)
6
>>> next(x)
Traceback (most recent call last):
File "<pyshell#48>", line 1, in <module>
next(x)
StopIteration
```

Learning Objectives

- Miscellaneous
  - Iterator
  - User-defined exception

Exception hierarchy: Built-in



```
BaseException
 +-- SystemExit
 +-- KeyboardInterrupt
 +-- GeneratorExit
 +-- Exception
      +-- StopIteration
      +-- ArithmeticError
      |    +-- FloatingPointError
      |    +-- OverflowError
      |    +-- ZeroDivisionError
      +-- AssertionError
      +-- AttributeError
      +-- BufferError
      +-- EOFError
      +-- ImportError
      +-- LookupError
      |    +-- IndexError
      |    +-- KeyError
      +-- MemoryError
      +-- NameError
      |    +-- UnboundLocalErro
      +-- OSError
      |    +-- BlockingIOError
      |    +-- ChildProcessError
      |    +-- ConnectionError
      |    |    +-- BrokenPipeError
      |    |    +-- ConnectionAbortedError
      |    |    +-- ConnectionRefusedError
      |    |    +-- ConnectionResetError
      |    +-- FileExistsError
      |    +-- FileNotFoundError
      |    +-- InterruptedError
      |    +-- IsADirectoryError
      |    +-- NotADirectoryError
      |    +-- PermissionError
      |    +-- ProcessLookupError
      |    +-- TimeoutError
      +-- ReferenceError
      +-- RuntimeError
      |    +-- NotImplementedError
      +-- SyntaxError
      |    +-- IndentationError
      |         +-- TabError
      +-- SystemError
      +-- TypeError
      +-- ValueError
      |    +-- UnicodeError
      |         +-- UnicodeDecodeError
      |         +-- UnicodeEncodeError
      |         +-- UnicodeTranslateError
      +-- Warning
           +-- DeprecationWarning
           +-- PendingDeprecationWarning
           +-- RuntimeWarning
           +-- SyntaxWarning
           +-- UserWarning
           +-- FutureWarning
           +-- ImportWarning
           +-- UnicodeWarning
           +-- BytesWarning
           +-- ResourceWarning
```

User-defined exception

- User-defined exceptions can be created by defining a new exception class, typically derived from the Exception class, either directly or indirectly
- Exception classes are usually kept simple, often only offering information about the error
- When creating a module that can raise several distinct errors, a common practice is to create a base class for exceptions defined by that module, and subclass that to create specific exception classes for different error conditions

```
class Error(Exception):
    """Base class for your module."""
    pass

class InputError(Error):
    """Exception error for input."""
    def __init__(self, expr, msg):
        self.expr = expr
        self.msg = msg

def checkSyntax(s):
    return False

try:
    x = input("Input your expr:")
    if checkSyntax(x) == False:
        raise InputError(x, “Error Information")
except InputError as e:
    print(e.msg, "in", x)
```

Mutability

Mutable vs Immutable?

Immutable objects are those whose value cannot be changed in place after assignment or initialization. They allocate new memory whenever their value is changed. Eg. immutable (int, float, tuple…), mutable (list, dict, set…)

Recap

- Miscellaneous
  - Iterator
  - User-defined exception

Readings (recommended)

- Python HOWTOs (if you haven’t read them)
  - Iterators
  - Generators
  - Generator expressions and list comprehensions