UNIVERSITÄT
BIELEFELD
Faculty of Technology

Programming

Object-oriented programming

Luna Pianesi

Faculty of Technology, Bielefeld University

Recap

- Functions: reusable blocks of code with defined syntax
- Variable scope: local vs global
- Debugging with `try`/`except` statements
- Functional programming: treating functions as objects
  - First-class functions
  - Recursions
  - Lambda functions
  - `map`, `reduce` and `filter`
  - List comprehensions, generator expressions
- Lazy evaluation

UNIVERSITÄT
BIELEFELD

Programming Errors & Debugging

Classes

Modules

Packages

Programming errors

Recognizing different types of errors:

- Syntactic: spelling & grammar mistakes
  - e.g. $avg = (x\,y)/2$
- Semantic: mistakes in meaning, context, or program flow
  - e.g. $avg = x + y/2$ or $avg = (x + z)/0$

Distinction between

- Compile-time errors (syntactic, semantic)
- Runtime errors (semantic)

UNIVERSITÄT BIELEFELD

RuntimeError

Changing the size of `my_dict` in loop

```python
# dictionary filled with arbitrary elements
my_dict = {'key': 'value', 1: 'text', (1, 2)
    : 'text'}

# for-loop over keys of my_dict with control
    variable 'key'
for key in my_dict:
    my_dict[(key, 1, 2, 3)] = 'new␣element'
```

Catching exceptions

Controlled treatment of anticipated exceptions:

```
while True:
    try:
        x = int(input("Please enter a number: "))
        break
    except ValueError:
        print("Oops!  That was no valid number.  Try again...")
```

UNIVERSITÄT BIELEFELD

Raising exceptions

Use `raise` keyword to throw exceptions:

```python
def myFunction(collection):

    if len(collection) == 0:
        raise RuntimeError("Invalid input: empty collection")
    # do something ..
    return

myFunction(list())
```

UNIVERSITÄT BIELEFELD

Raising exceptions

Check properties of input parameters using the `assert` statement:

```python
def myFunction(collection):

    assert len(collection) > 0, "Invalid␣input:␣empty␣collection"

    # do something ..
    return

myFunction(list())
```

Failed assertions result in an `AssertionError`

UNIVERSITÄT BIELEFELD

Debugging

PDB—the Python debugger

- Enables step-by-step proceeding of statements in Python programs
- Interaction with Python program at runtime
- Debugger is invoked by *breakpoints*
- Set breakpoint in arbitrary location of your code by
  - calling builtin “`breakpoint()`” function (Python version ≥ 3.7)
  - statement “`import pdb; pdb.set_trace()`”

Python debugger—example

```python
# dictionary filled with arbitrary elements
my_dict = {'key': 'value', 1: 'text', (1, 2)
   : 'text'}

# invoke Python debugger
breakpoint()

# for-loop over keys of my_dict with control
    variable 'key'
for key in my_dict:
    my_dict[(key, 1, 2, 3)] = 'new␣element'
```

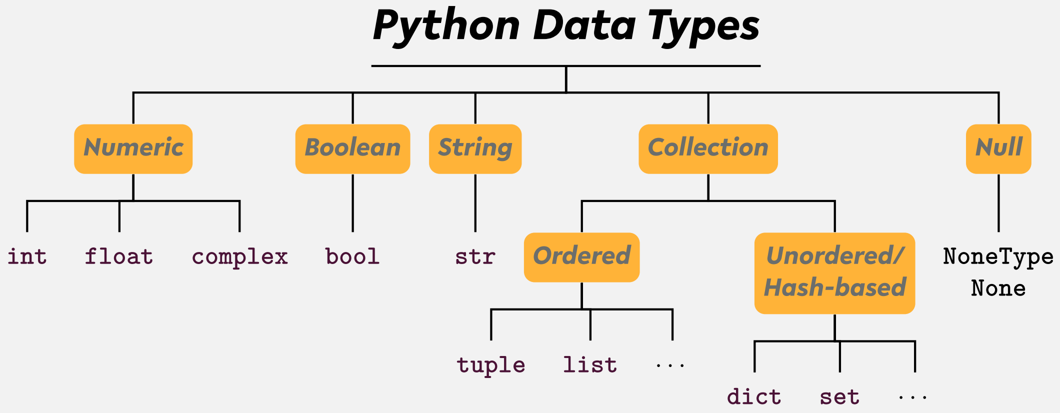
UNIVERSITÄT
BIELEFELD

Programming Errors & Debugging

Classes

Modules

Packages



… and user-defined types!

Creating new types

- A `class` defines a new type
- It can provide
  - class variables & functions
  - instance variables & functions

UNIVERSITÄT BIELEFELD

Classes—example of code reuse

```python
class Library:
    description = 'This␣is␣a␣Library'

    def __init__(self, name):
        # name the library
        self.name = name
        # create empty book storage on initialization
        self.storage = list()

    def addBook(self, book):
        self.storage.append(book)

    def getAllBooks(self):
        return tuple(self.storage)

myLib = Library('Bodleian␣Library')
myLib.addBook('The␣Art␣of␣Computer␣Programming␣(D.␣Knuth)')
```

Programming Errors & Debugging

Classes

Modules

Packages

UNIVERSITÄT BIELEFELD

Modules

- Every `.py` file is a module
- Modules can host functions, variables, and classes
- Imported modules with `import` statement
- Should not have blocks of code that are immediately executed
- Explicit reference to module scope: `global`
- Name of module available as global variable `__name__`

Modules—example of code reuse

mystringutils.py

```python
#
# A module for all kinds of string utils
#

def findSubstringInStrings(stringCollection,
      pattern):
    occ = list()
    for i, s in enumerate(stringCollection):
        j = s.find(pattern)
        while j != -1:
            occ.append((i, j))
            j = s.find(pattern, j+1)
    return occ
```

myscript.py

```python
#!/usr/bin/env python3

import mystringutils

if __name__ == '__main__':
    myStringList = ['the␣rain␣in␣spain',
        'ain\'t␣no␣sunshine',
        'she␣was␣greeted␣with␣disdain']

    occOfAin = mystringutils.
        findSubstringInStrings(myStringList,
         'ain')
    print(occOfAin)
```

UNIVERSITÄT
BIELEFELD

Programming Errors & Debugging

Classes

Modules

Packages

UNIVERSITÄT BIELEFELD

Packages

- Way of structuring multiple modules into a directory hierarchy
- Package directories must contain a `__init__.py` file
- Can be imported the same way as modules
- Python itself offers many packages, and even more third-party packages are available through *package managers* such as `conda`

Quiz

- In Python, a class is ____________ for an object.

  a nuisance   an instance   a blueprint   a distraction

- Consider the following class:

```python
class Dog:
    def __init__(self, name, age):
        self.name = name
        self.age = age
```

  What is the correct statement to instantiate a `Dog` object?

  - `Dog('Rufus', 3)`
  - `Dog(self, 'Rufus', 3)`
  - `Dog.__init__('Rufus', 3)`

source (in part): `https://realpython.com/quizzes`

UNIVERSITÄT BIELEFELD

Quiz

- In Python, a class is ____________ for an object.

  a nuisance an instance [✓ a blueprint] a distraction

- Consider the following class:

```python
class Dog:
    def __init__(self, name, age):
        self.name = name
        self.age = age
```

What is the correct statement to instantiate a `Dog` object?

  - [✓ `Dog('Rufus', 3)`]
  - `Dog(self, 'Rufus', 3)`
  - `Dog.__init__('Rufus', 3)`

source (in part): `https://realpython.com/quizzes`

UNIVERSITÄT
BIELEFELD

Recap

UNIVERSITÄT
BIELEFELD

Summary

- Compile-time and runtime errors
- Python debugger, a tool for hunting runtime errors (bugs)
- Code reuse through
  - Functions
  - Classes
  - Modules & Packages

UNIVERSITÄT BIELEFELD

What comes next?

- Write your first class, module, and Python script
- Due date for this week's exercises is ***Wednesday, November 27, 2pm, 2024***.

Next lecture: Data management & analysis, Jupyter Notebook, text mining ...