NPRG065: Programming in Python *Lecture 5*

http://d3s.mff.cuni.cz

Department of Distributed and Dependable Systems

Tomas Bures

Petr Hnetynka

{bures,hnetynka}@d3s.mff.cuni.cz

CHARLES UNIVERSITY IN PRAGUE

faculty of mathematics and physics

Basic I/O and Exceptions (cont.)

Handling exceptions

- Reminder

```
import sys

try:
    f = open(sys.argv[1], 'r')
except OSError:
    print('cannot open', sys.argv[1])
else:
    print('File has', len(f.readlines()), 'lines')
    f.close()
```

`with`

- partially similar to Java's "try with resources" or C#'s **`with`**
  - calls **`close()`** but does not handle exceptions
- usable not only with files
  - will be covered later

```
with open('workfile') as f:
    read_data = f.read()
    // do something with read data
print(f.closed)  // prints true
```

Examine and run basic_io.py

Raising exceptions

- raise

```
raise NameError('HiThere')
```

```
raise ValueError
```

- Exceptions can be re-raised

```
try:
    raise NameError('HiThere')
except NameError:
    print('An exception flew by!')
    raise
```

Department of Distributed and Dependable Systems

Own exceptions

- exception ~ an instance of a class extending the Python's built-in **`Exception`** class
  - classes, extending, etc. will be covered the next lecture

```
class MyException(Exception):
    pass

try:
    raise MyException
except:
    print('Exception occurred')
```

See
own_exception.py

Functions and their parameters

Department of Distributed and Dependable Systems

Functions

-
  ```
  def function_name(parameters):
      body
      return value    # optional
  ```

- Are first-class entities
  - e.g., can be assigned or passed as arguments

```
def say_hello():
    print('Hello world')

say_hello()

print_hello = say_hello

print_hello()
```

Functions

- Five kinds of parameters
  - positional-or-keyword – most common and default variant
    - `def func(foo, bar=None):`
  - positional-only – used only in several builtin functions
  - keyword-only
    - `def func(arg, *, kw_only1, kw_only2):`
  - var-positional – an arbitrary sequence of positional arguments
    - `def func(*args, **kwargs):`
  - var-keyword – an arbitrary sequence of keywords arguments
    - `def func(*args, **kwargs):`
- Parameters – passing by-value

Examine and run functions.py

Functions

- Functions can be defined in functions
  - e.g., to hide implementation

```
def factorial(number):
    # error handling
    if not number >= 0:
        return -1

    def inner_factorial(number):
        if number <= 1:
            return 1
        return number*inner_factorial(number-1)
    return inner_factorial(number)
```

Examine and run functions.py

Functions and visibility

- Visibility of variables in function is as usual

```
def outer():
    test = 1
    def inner():
        test = 2
        print('  inner:', test)
    inner()
    print('  outer:', test)


test = 0  # global scope
outer()
print('  global:', test)
```

Examine and run functions.py

Functions and visibility

- But we can access variables in different scope

test variable from the nearest enclosing scope

```
nonlocal test


global test
```

test variable from the global scope

Examine and run functions.py