UNIVERSITÄT BIELEFELD
Faculty of Technology

Programming

Functions, debugging, functional programming and lazy evaluation

Luna Pianesi

Faculty of Technology, Bielefeld University

Functions

Debugging

Functional programming

Lazy evaluation

UNIVERSITÄT BIELEFELD

Functions

```
def «functionName» ( «parameterName1», «parameterName2», ... ):
    «statement»
    return «statement»
```

Mind the indentation!

gray = optional

Variable Scope

Functions have a separate variable scope!

- internal variables are not accessible from outside
- calling global functions and variables is possible
  - Reading global variables is discouraged
- Changing global variables requires

  «`global variableName`»

  statement inside function (highly discouraged)

source: https://www.learnpython.org/en/Functions

Functions—a simple example

```
1 def myFirstFunction():
2     print('this␣is␣my␣first␣function')
3
4 # call function
5 myFirstFunction()
6
7 # save return value in variable
8 hereComesNothing = myFirstFunction()
```

Functions—example of code reuse

```
1  def findSubstringInStrings(stringCollection, pattern):
2      occ = list()
3      for i, s in enumerate(stringCollection):
4          j = s.find(pattern)
5          while j != -1:
6              occ.append((i, j))
7              j = s.find(pattern, j+1)
8      return occ
9
10 myStringList = ['the␣rain␣in␣spain', 'ain\'t␣no␣sunshine',
11     'she␣was␣greeted␣with␣disdain']
12
13 occOfAin = findSubstringInStrings(myStringList, 'ain')
```

Quiz

Have you ever seen a function calling itself? Consider the following:

```
1 def fun(x):
2     if len(x) > 1:
3         return fun(x[1:])
4     return x
```

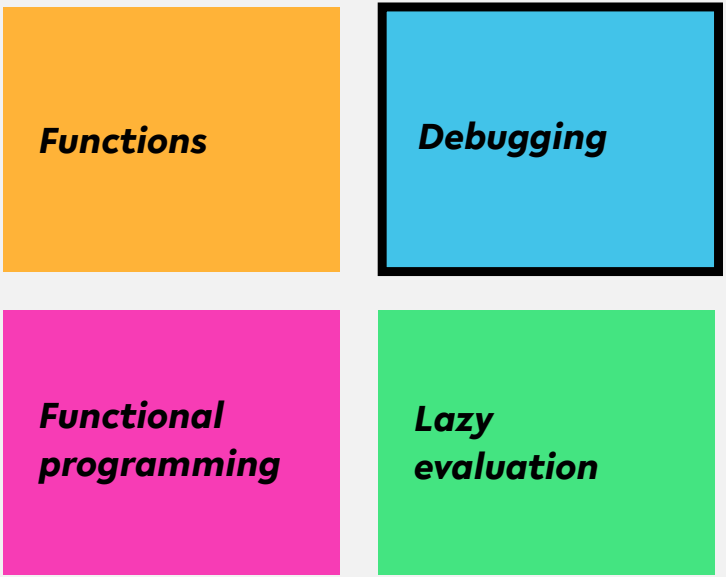
What does the function call `fun([1,2,3,4])` return?

Quiz

Have you ever seen a function calling itself? Consider the following:

```
1 def fun(x):
2     if len(x) > 1:
3         return fun(x[1:])
4     return x
```

What does the function call `fun([1,2,3,4])` return? [4]


Functions
Debugging
Functional programming
Lazy evaluation

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)

RuntimeError

Changing the size of `my_dict` in loop

```
1 # dictionary filled with arbitrary elements
2 my_dict = {'key': 'value', 1: 'text', (1, 2)
    : 'text'}
3
4 # for-loop over keys of my_dict with control
    variable 'key'
5 for key in my_dict:
6     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...")
```

Raising exceptions

Use `raise` keyword to throw exceptions:

```
1  def myFunction(collection):
2
3      if len(collection) == 0:
4          raise RuntimeError("Invalid input: empty collection")
5      # do something ..
6      return
7
8  myFunction(list())
```

Raising exceptions

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

```
1  def myFunction(collection):
2
3      assert len(collection) > 0, "Invalid input: empty collection"
4
5      # do something ..
6      return
7
8  myFunction(list())
```

Failed assertions result in an `AssertionError`

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 $\geq$ 3.7)
  - statement “`import pdb; pdb.set_trace()`”

Python debugger—example

```
1 # dictionary filled with arbitrary elements
2 my_dict = {'key': 'value', 1: 'text', (1, 2)
    : 'text'}
3
4 # invoke Python debugger
5 breakpoint()
6
7 # for-loop over keys of my_dict with control
    variable 'key'
8 for key in my_dict:
9     my_dict[(key, 1, 2, 3)] = 'new␣element'
```

Quiz

- Is improper indentation a syntactic or semantic error?
- Consider the following code:

```
1 def str2int(x):
2     try:
3         return int(x)
4     _____________ ValueError:
5         return -1
```

What keyword should be used here?

`except` `raise` `else` `Exception`

source: https://quizizz.com/

Quiz

- Is improper indentation a syntactic or semantic error? syntactic
- Consider the following code:

```
1 def str2int(x):
2     try:
3         return int(x)
4     _____________ ValueError:
5         return -1
```

What keyword should be used here?

except✔ raise else Exception

source: https://quizizz.com/

Functions

Debugging

Functional programming

Lazy evaluation

Functional programming

See Jupyter Notebook!

- Declarative programming paradigm
- Result constructed by applying and composing functions

Functions

Debugging

Functional programming

Lazy evaluation

Lazy evaluation

See Jupyter Notebook!

- Concept in functional programming
- Refers to how function arguments are processed ***only*** when an expression is being evaluated

Recap

Summary

- Code reuse through functions
- Compile-time and runtime errors
- Python debugger, a tool for hunting runtime errors (bugs)
- Declarative paradigm: functional programming
- Lazy evaluation and implicit Boolean conversion

What comes next?

- Write your first function
- Familiarize yourself with the Python Debugger
- Due date for this week's exercises is ***Wednesday, November 13, 2pm, 2024***.

Next lecture: Object-oriented programming … …