UPPSALA
UNIVERSITET

Code Examples

2017-05-22

Advanced Scientific Programming with Python

Adapted from:
https://realpython.com/blog/python/primer-on-python-decorators/

Nested Functions In Python

- In Python it is possible to define functions inside functions

```
def parent():
    print("Printing from the parent() function.")

    def first_child():
        return "Printing from the first_child() function."

    def second_child():
        return "Printing from the second_child() function."

    print(first_child())
    print(second_child())
```

- The child functions are only available in the scope of the parent function

```
>>> parent()
Printing from the parent() function.
Printing from the first_child() function.
Printing from the second_child() function
```

Returning A Function From Other Functions

- The return value of a function can be another function

```
def parent(num):
    def first_child():
        return "Printing from the first_child() function."
    def second_child():
        return "Printing from the second_child() function."

    try:
        assert num == 10
        return first_child
    except AssertionError:
        return second_child

>>> foo = parent(10)
>>> bar = parent(11)

>>> print(foo())
>>> print(bar())
Printing from the first_child() function.
Printing from the second_child() function.
```

Decorators In Python

- A decorator is a wrapper around a function

```
def my_decorator(some_function):

    def wrapper():

        print("Something is happening before some_function() is called.")

        some_function()

        print("Something is happening after some_function() is called.")

    return wrapper


def just_some_function():
    print("Wheee!")


>>> just_some_function = my_decorator(just_some_function)
>>> just_some_function()
Something is happening before some_function() is called.
Wheee!
Something is happening after some_function() is called.
```

Decorators In Python

- Simplified syntax using the @ for decorators

```
def my_decorator(some_function):

    def wrapper():

        print("Something is happening before some_function() is called.")

        some_function()

        print("Something is happening after some_function() is called.")

    return wrapper

@my_decorator
def just_some_function():
    print("Wheee!")
```

```
>>> just_some_function()
Something is happening before some_function() is called.
Wheee!
Something is happening after some_function() is called.
```

Data visualisation in Python

Data Visualisation Tools In Python

- matplotlib (http://matplotlib.org/)
- seaborn (https://seaborn.pydata.org/)
- Bokeh (http://bokeh.pydata.org/en/latest/)
- Plotly (https://plot.ly/python/)