User-defined functions

PYTHON DATA SCIENCE TOOLBOX (PART 1)

Hugo Bowne-Anderson
Instructor

datacamp

You'll learn:

- Define functions without parameters
- Define functions with one parameter
- Define functions that return a value
- Later: multiple arguments, multiple return values

Built-in functions

- str()

```
x = str(5)

print(x)
```

```
'5'
```

```
print(type(x))
```

```
<class 'str'>
```

Defining a function

```python
def square():     # <- Function header
    new_value = 4 ** 2    # <- Function body
    print(new_value)
square()
```

```
16
```

Function parameters

```python
def square(value):
    new_value = value ** 2
    print(new_value)

square(4)
```

```
16
```

```python
square(5)
```

```
25
```

5. Function parameters
What if you wanted to square any other number besides 4, though? To add that functionality, you add a parameter to the function definition in between the parentheses. Here you see that we've added a parameter value and in the new function body, the variable new_value takes the square of value, which is then printed out. We can now square any number that we pass to the function square as an argument. A quick word on parameters and arguments: when you define a function, you write parameters in the function header. When you call a function, you pass arguments into the function.

Return values from functions

- Return a value from a function using return

```
def square(value):
    new_value = value ** 2
    return new_value
num = square(4)

print(num)
```

```
16
```

6. Return values from functions
The function square now accepts a single parameter and prints out its squared value. But what if we don't want to print that value directly and instead we want to return the squared value and assign it to some variable? You can have your function return the new value by adding the return keyword, followed by the value to return. Now we can assign to a variable num the result of the function call as you see here.

Docstrings

- Docstrings describe what your function does
- Serve as documentation for your function
- Placed in the immediate line after the function header
- In between triple double quotes """

```python
def square(value):
    """Return the square of a value."""
    new_value = value ** 2
    return new_value
```

Let's practice!

PYTHON DATA SCIENCE TOOLBOX (PART 1)

Multiple Parameters and Return Values

PYTHON DATA SCIENCE TOOLBOX (PART 1)

Hugo Bowne-Anderson
Instructor

Multiple function parameters

- Accept more than 1 parameter:

```python
def raise_to_power(value1, value2):
    """Raise value1 to the power of value2."""
    new_value = value1 ** value2
    return new_value
```

- Call function: # of arguments = # of parameters

```python
result = raise_to_power(2, 3)

print(result)
```

```
8
```

A quick jump into tuples

- Make functions return multiple values: Tuples!
- Tuples:
    - Like a list - can contain multiple values
    - Immutable - can't modify values!
    - Constructed using parentheses ()

```
even_nums = (2, 4, 6)


print(type(even_nums))
```

```
<class 'tuple'>
```

Unpacking tuples

- Unpack a tuple into several variables:

```
even_nums = (2, 4, 6)

a, b, c = even_nums
```

```
print(a)
```

```
2
```

```
print(b)
```

```
4
```

```
print(c)
```

```
6
```

Accessing tuple elements

- Access tuple elements like you do with lists:

```
even_nums = (2, 4, 6)

print(even_nums[1])
```

```
4
```

```
second_num = even_nums[1]

print(second_num)
```

```
4
```

- Uses zero-indexing

Returning multiple values

```python
def raise_both(value1, value2):
    """Raise value1 to the power of value2
    and vice versa."""

    new_value1 = value1 ** value2
    new_value2 = value2 ** value1

    new_tuple = (new_value1, new_value2)

    return new_tuple
```

```python
result = raise_both(2, 3)

print(result)
```

```
(8, 9)
```

6. Returning multiple values
Let's now modify the behavior of your raise function. Instead of returning just the value of value1 raised to the power of value2, let's also return the value of value2 raised to the power of value1. You thus need to make raise return two values instead of one. We can use what we now know of tuples to do this! We first change the name of our function and the docstring to reflect the new behavior of our function. We then, in the function body, construct a tuple consisting of the values we want the function to return and, also in the function body, we return the tuple! Calling the function constructed demonstrates that it does exactly what we want!

Let's practice!

PYTHON DATA SCIENCE TOOLBOX (PART 1)

Bringing it all together

PYTHON DATA SCIENCE TOOLBOX (PART 1)

Hugo Bowne-Anderson
Instructor

datacamp

You’ve learned:

- How to write functions
    - Accept multiple parameters
    - Return multiple values
- Up next: Functions for analyzing Twitter data

Basic ingredients of a function

- Function Header

```
def raise_both(value1, value2):
```

- Function body

```
    """Raise value1 to the power of value2
    and vice versa."""

    new_value1 = value1 ** value2
    new_value2 = value2 ** value1

    new_tuple = (new_value1, new_value2)

    return new_tuple
```

Let's practice!

PYTHON DATA SCIENCE TOOLBOX (PART 1)

Congratulations!

PYTHON DATA SCIENCE TOOLBOX (PART 1)

Hugo Bowne-Anderson
Instructor

datacamp

Next chapters:

- Functions with default arguments
- Functions that accept an arbitrary number of parameters
- Nested functions
- Error-handling within functions
- More function use in data science!

Let's practice!

PYTHON DATA SCIENCE TOOLBOX (PART 1)