PYTHON DATA SCIENCE TOOLBOX I

User-defined functions

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()`

```
In [1]: x = str(5)

In [2]: print(x)
'5'

In [3]: print(type(x))
<class 'str'>
```

Defining a function

```
In [1]: def square():
   ...:     new_value = 4 ** 2
   ...:     print(new_value)

In [2]: square()
16
```

Function header

Function body

(Indented)

Function parameters

```
In [1]: def square(value):                parameter
   ...:     new_value = value ** 2
   ...:     print(new_value)

In [2]: square(4)                argument
16

In [3]: square(5)
25
```

Return values from functions

- Return a value from a function using `return`

```
In [1]: def square(value):
   ...:     new_value = value ** 2
   ...:     return new_value

In [12]: num = square(4)

In [13]: print(num)
16
```

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 """

```
In [1]: def square(value):
   ...:     """Return the square of a value."""
   ...:     new_value = value ** 2
   ...:     return new_value
```

Let’s practice!

PYTHON DATA SCIENCE TOOLBOX I

Multiple parameters and return values

Multiple function parameters

- Accept more than 1 parameter:

```
In [1]: 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

```
In [2]: result = raise_to_power(2, 3)

In [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 ( )

```
In [1]: even_nums = (2, 4, 6)

In [2]: print(type(even_nums))
<class 'tuple'>
```

Unpacking tuples

- Unpack a tuple into several variables:

```
In [1]: even_nums = (2, 4, 6)

In [2]: a, b, c = even_nums

In [3]: print(a)
2

In [4]: print(b)
4

In [5]: print(c)
6
```

Accessing tuple elements

- Access tuple elements like you do with lists:

```
In [1]: even_nums = (2, 4, 6)

In [2]: print(even_nums[1])
4

In [3]: second_num = even_nums[1]

In [4]: print(second_num)
4
```

- Uses zero-indexing

Returning multiple values

raise.py

```
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
```

```
In [1]: result = raise_both(2, 3)

In [2]: print(result)
(8, 9)
```

Let’s practice!

Bringing it all together

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

raise.py

```
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
```

Function header

Function body

Let’s practice!

Congratulations!

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!

See you in the next chapter!