Functions & Libraries

Jannusch Bigge

14.11.2023

Functions

Functions

Often we want to do the same thing multiple times.

Functions

Often we want to do the same thing multiple times.
But we don’t want to write the same code multiple times.

Functions

Often we want to do the same thing multiple times.
But we don't want to write the same code multiple times.

Solution: Functions

- Reusable code

Functions

Often we want to do the same thing multiple times.
But we don't want to write the same code multiple times.

Solution: Functions

- Reusable code
- Easier to read

Functions

Often we want to do the same thing multiple times.
But we don't want to write the same code multiple times.

Solution: Functions

- Reusable code
- Easier to read
- Easier to debug

Functions - Definition in Python

Definition:

```
def function_name(arguments):
    # do something
    return something
```

Functions - Definition in Python

Definition:

```
def function_name(arguments):
    # do something
    return something
```

- **def** *name*(*arguments*): - Start of the function

Functions - Definition in Python

Definition:

```python
def function_name(arguments):
    # do something
    return something
```

- **def** *name*(*arguments*): - Start of the function
- **return** *something* - End of the function

Functions - Calling

Define:

```
def fibbonacci(number):
    a = 0
    ...
        b = a + b
    return a
```

Calling the function:

```
>>> result = fibbonacci(6)
>>> print(result)
8
```

Some examples

You allready know some functions:

- **print**(*something*)

Some examples

You allready know some functions:

- **print**(*something*)
- **len**(*something*)

Some examples

You allready know some functions:

- **print**(*something*)
- **len**(*something*)
- **range**(*something*)

Some examples

You allready know some functions:

- **print**(*something*)
- **len**(*something*)
- **range**(*something*)
- **input**(*something*)

Return values

You can return none, one or multiple values.

Return values

You can return none, one or multiple values.

- **None** - Nothing

Return values

You can return none, one or multiple values.

- **None** - Nothing
- **return a** - One value

How to access multiple return values?

Return values

You can return none, one or multiple values.

- **None** - Nothing
- **return a** - One value
- **return a, b, c** - Multiple values

How to access multiple return values?

Tuples

Special data type in Python: **Tuple**

Tuples

Special data type in Python: **Tuple**
Stores multiple values in one variable.

Tuples

Special data type in Python: **Tuple**
Stores multiple values in one variable.

- Immutable
- Ordered
- Can contain multiple data types

Tuples - Examples

```
>>> my_tuple = (1, 2, 3)
>>> print(my_tuple)
(1, 2, 3)
>>> print(my_tuple[0])
1
>>> print(my_tuple[1])
2
>>> print(my_tuple[2])
3
```

Stuff like **len()** and **for** works as expected.

Return types

What is returned by the function?

Return types

What is returned by the function?

- In python you are not forced to reveal that.

Return types

What is returned by the function?

- In python you are not forced to reveal that.

Why?

Return types

What is returned by the function?

- In python you are not forced to reveal that.

Why?

- Python is a so called dynamically typed language.

Return types

What is returned by the function?

- In python you are not forced to reveal that.

Why?

- Python is a so called dynamically typed language.
- You also don't have to specify the type of the arguments.

Return types

What is returned by the function?

- In python you are not forced to reveal that.

Why?

- Python is a so called dynamically typed language.
- You also don't have to specify the type of the arguments.

Should I do it anyway?

Return types

What is returned by the function?

- In python you are not forced to reveal that.

Why?

- Python is a so called dynamically typed language.
- You also don't have to specify the type of the arguments.

Should I do it anyway?

- There is a reason why Python has this built in functionality.

Return types

What is returned by the function?

- In python you are not forced to reveal that.

Why?

- Python is a so called dynamically typed language.
- You also don't have to specify the type of the arguments.

Should I do it anyway?

- There is a reason why Python has this built in functionality.

Type definition

Defining the return type:

Type definition

Defining the return type:

```
def fibbonacci(number) -> int:
    a = 0
    ...
        b = a + b
    return a
```

Type definition

Defining the return type:

```
def fibbonacci(number) -> int:
    a = 0
    ...
        b = a + b
    return a
```

Defining the argument type:

Type definition

Defining the return type:

```
def fibbonacci(number) -> int:
    a = 0
    ...
        b = a + b
    return a
```

Defining the argument type:

```
def fibbonacci(number: int):
    a = 0
    ...
        b = a + b
    return a
```

Type definition

I don't know the type:

Type definition

I don't know the type:

```
print(type(something))
```

Type definition

I don't know the type:

```
print(type(something))
```

For more complex returns you can use the typing module:

Type definition

I don't know the type:

```
print(type(something))
```

For more complex returns you can use the typing module:

- You can use **typing.Any** to indicate that you don't know the type.

Type definition

I don't know the type:

```python
print(type(something))
```

For more complex returns you can use the typing module:

- You can use **typing.Any** to indicate that you don't know the type.
- You can use **typing.Union** to indicate that you don't know the type but it is one of the types you specified.

Type definition

I don't know the type:

```
print(type(something))
```

For more complex returns you can use the typing module:

- You can use **typing.Any** to indicate that you don't know the type.
- You can use **typing.Union** to indicate that you don't know the type but it is one of the types you specified.
- You can use **typing.Optional** to indicate that the type is one of the types you specified or **None**.

Type definition

I don't know the type:

```
print(type(something))
```

For more complex returns you can use the typing module:

- You can use **typing.Any** to indicate that you don't know the type.
- You can use **typing.Union** to indicate that you don't know the type but it is one of the types you specified.
- You can use **typing.Optional** to indicate that the type is one of the types you specified or **None**.

Now we know how to write and use functions.

Now we know how to write and use functions.

Let's start using code from other people.

Libraries

Libraries

Many libraries solve a lot of problems.

Libraries

Many libraries solve a lot of problems.

- **math** - Math functions
- **secrets** - strong random numbers
- **numpy** - fast/complex math
- **matplotlib** - plotting
- **pandas** - data analysis
- **tensor-flow** - machine learning

Libraries

To use a library you have to import it.

Libraries

To use a library you have to import it.

```
import math
```

Libraries

To use a library you have to import it.

```
        import math
```

Now you can use the functions from the library.

Libraries

To use a library you have to import it.

```
import math
```

Now you can use the functions from the library.

```
print(math.sqrt(4))
2.0
```

Sometimes you only want to import a single function.

Libraries

To use a library you have to import it.

```
import math
```

Now you can use the functions from the library.

```
print(math.sqrt(4))
2.0
```

Sometimes you only want to import a single function.

```
from math import sqrt
print(sqrt(4))
2.0
```

Documentation

In general you can find the documentation of a library on the internet.

Documentation

In general you can find the documentation of a library on the internet.

- **math** - https://docs.python.org/3/library/math.html
- **tensor-flow** - https://www.tensorflow.org/api_docs/python/tf

Libraries

pip

Some libraries are not installed by default.
You have to install them first.

Libraries

pip

Some libraries are not installed by default.

You have to install them first.

But we will talk about that next week.

Next week:
More data types and a bigger task