CSCI 141

Scott Wehrwein

Defining Functions
Goals

• Know the syntax for defining your own functions

• Know how to define and use functions that take no arguments and return no values
Functions, Revisited

• We’ve been using functions since day 1:

    \text{print}("Hello, World!")

• Built-in functions so far:
  \text{print}, \text{input}, \text{type}, \text{len}, \text{int}, \text{str}, \ldots

• We can import more functions:
  \text{import math}
  \text{import turtle}
  \text{math.sqrt}(4)
  \text{turtle.Turtle}()
Functions, Revisited

What is a function, anyway?

It’s a chunk of code with a name.
• It may take arguments as input
• It may do something that has an effect
• It may return a value

```
print( "Hello world" )
```

Input(s):
• 0 or more values
• (optional) sep and end keywords

Return value:
• none

Effects: prints arguments to the screen, with given separator and end
Functions, Revisited

What is a function, anyway?

It’s a chunk of code with a name.
• It may take arguments as input
• It may do something that has an effect
• It may return a value

```python
input(“Enter a number:”)
```

Input(s):
• none, or
• a string to print as a prompt

Return value:
• the input from the user

Effects: prompts for user input and reads it from the keyboard
Functions, Revisited

What is a function, anyway?

It’s a chunk of code with a name.
• It may take arguments as input
• It may do something that has an effect
• It may return a value

Input(s):
• a value

Return value:
• the type of the value

type (6/2)

Effects: none
Functions, Revisited

What is a function, anyway?

It’s a chunk of code with a name.
- It may take arguments as input
- It may do something that has an effect
- It may return a value

Input(s):
- a number

Return value:
- the sine of the value

Effects: none

\[ \text{math.sin}(\text{math.pi}/2) \]
Functions, Revisited

What **is** a function, anyway?

It’s a chunk of code with a name.
• It *may* take **arguments** as input
• It *may* do something that has an effect
• It *may* return a value

Input(s):
• a number

Return value:
• none

Effects: moves the turtle forward by the given number of units
What **is** a function, anyway?

- So far we’ve treated functions as “**black boxes**”, code someone else wrote that does stuff for us.
- All we know are the inputs, effects, and return value.
- We don’t know how it’s done.

\[ \text{Input(s)} \rightarrow \text{black box} \rightarrow \text{Return value} \]

**(Effects)**

This is a **great** situation to be in!

A bunch of (potentially complicated), powerful stuff is wrapped up in a nice, easy-to-use package.
What if

You want a nice easy-to-use function that does something complicated, but nobody else has written it for you…

Now, you will have the power to write your own functions.
Writing Functions: Syntax

```python
def name(parameters):
    statements
```

Two important questions:
1. How does the function use its arguments (inputs)?
2. How does the function return a value (output)?

Let’s dodge these questions for a moment…
Functions: the simplest kind

No arguments, no return value:

```python
def name():
    statements
```

Example:

```python
def print_hello():
    print("Hello, world!")
```
The `print_hello` function

**Input(s):**
- none

**Return value:**
- none

**Effects:** prints "Hello" to the screen
Demo

• hello_fn.py
Demo: hello_fn.py

• define print_hello function

• The definition does nothing except make the function exist

• call it using print_hello()

• you can call it whenever/however many times

• except you can’t call it before it’s defined