Functions

Functions

Reusable pieces of code

- Built-in functions:
  - Common: print, type ...
  - Type: int, bool, float ...
  - Math: abs, pow, sum, max ...
  - list of built-in functions

Build your own functions

- Create your own functions
- Work like build-in functions
- Input variables → return statement
  - Each path through the program should hit a `return` statement

Style-guide

- use **2 or 4 spaces** (instead of tabs) for indentation → colab does this automatically
- limit line length to 78 characters
- naming conventions: use `CamelCase` for classes and `lowercase_with_underscores` for functions and variables
    a. Stay consistent with naming style
- place imports at the top of the file
- keep function definitions together
- keep top level statements, including function calls, together at the bottom of the program

Demo

1. Built-in
2. Simple functions
3. Parameters (inputs)

Libraries

- Import a library
- Adds more functionality

Examples:

1. Python standard library
2. math
3. sys
4. SciPy

Debugging

- Strategically use print()
- Testing of code

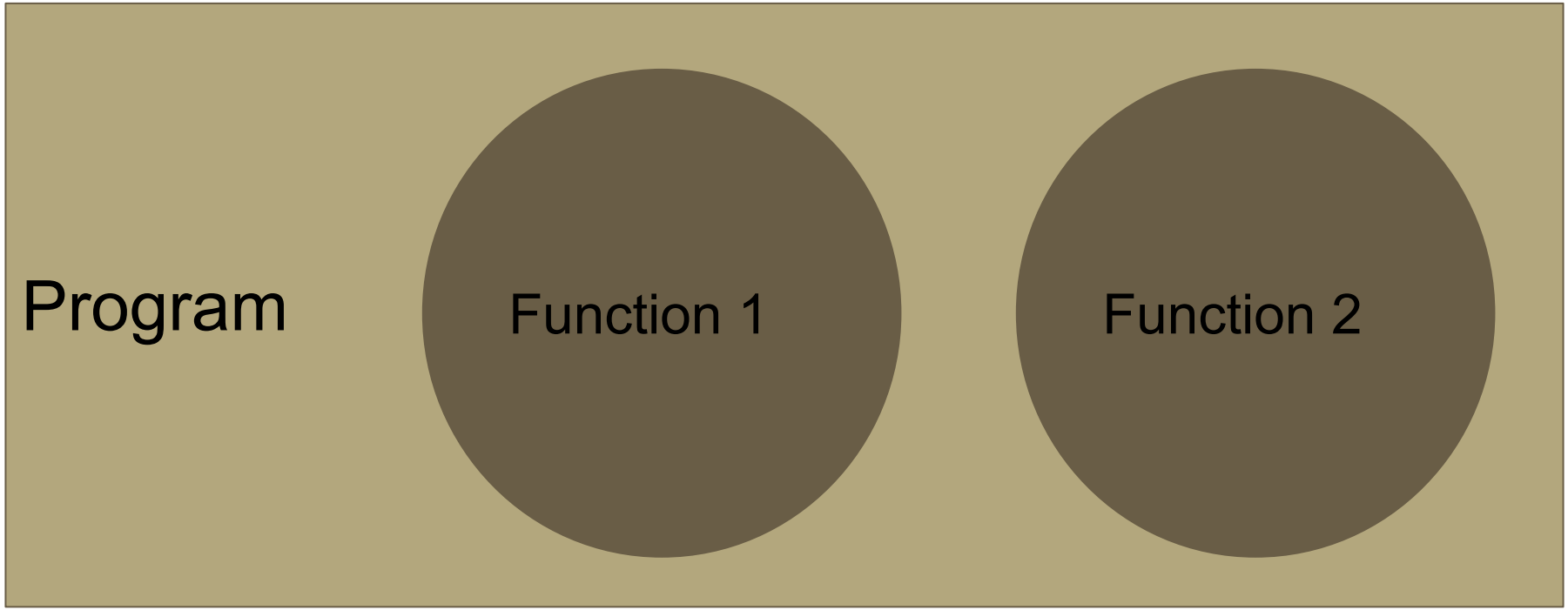
Unit-testing

- Used to automatically verify correctness of code
- Usually used to verify functions → known input and output
- Use:
  - import unittest
- Helpful with edge cases

Demo

1. Libraries
2. Debugging
3. Unit testing

Local vs Global Variables



Main () -function

- Used as the command station function
- All other functionality is controlled here

Syntax:

```
def main():

  print("Hello World!")

main()

print("Hello again!")
```

Demo

1. Global variable
2. Local variable
3. Main function

In-class assignment 4

1. Function practice
- Import the math library → make sure you're looking at python 3.7
- Create a function that gives you the:
  - Ceiling
  - Floor
  - Absolute value
- of a number given by the user and prints them out

2. Function practice
- Import the math library
- Create a function that receives a degree and changes it to radians
  - It calculates the sin/cos/tan for the new number and prints them out

Assignment 4

1. Using the math library, create a Python program to calculate the side of a right triangle. Write three functions: 1. SOH, 2. CAH. 3. TOA.
    a. Create a main() -function
    b. Create 3 functions for calculations
    c. Use user input to specify which side you're calculating
    d. Ask the user for input angle and side for the calculation
    e. Make continuous until use of "Exit" or "0" when asking for the calculated side
    f. How to calculate: link
2. Use the given unit tests to test out the solution of the functions created in problem (1). Add two (2) new unit tests to test out the main() function and two (2) unit tests for calculations.
    a. The purpose is to come up with tests that will see if the system will fail.
    b. If it fails, the program should be modified to make sure passes all tests.