CSc 110, Autumn 2017

Lecture 3: Functions

WHEN I STARTED PROGRAMMING, WE DIDN'T HAVE ANY OF THESE SISY "ICONS" AND "WINDOWS."

ALL WE HAD WERE ZEROS AND ONES -- AND SOMETIMES WE DIDN'T EVEN HAVE ONES.

I WROTE AN ENTIRE DATABASE PROGRAM USING ONLY ZEROS.

YOU HAD ZEROS? WE HAD TO USE THE LETTER "O."
Structure of a program

• No code should be placed outside a function. Instead use a `main` function.

  • The one exception is a call to your main function

```python
def main():
    message1()
    message2()
    print("Done with everything.")

def message1():
    print("This is message1.")

def message2():
    print("This is message2.")
    message1()
    print("Done with message2.")

main()
```
Functions question

• Write a program to print these figures using functions.
Development strategy

First version (unstructured):

- Create an empty program.
- Copy the expected output into it, surrounding each line with `print` syntax.
- Run it to verify the output.
Program version 1

def main():
    print("  ______")
    print(" /\"")
    print("/    ")
    print(" \______")
    print()    
    print(" /\"")
    print("/    ")
    print(" | STOP |")
    print(" \______")
    print()    
    print("  ______")
    print(" /\"")
    print("/    ")
    print("+-+")
    print("+-+")

main()
When to use functions (besides `main`)

- Place statements into a function if:
  - The statements are related structurally, and/or
  - The statements are repeated.

- You should not create functions for:
  - An individual `print` statement.
  - Only blank lines.
  - Unrelated or weakly related statements.
    (Consider splitting them into two smaller functions.)
Development strategy 2

Second version (structured, with redundancy):

- Identify the structure of the output.
- Divide the code into functions based on this structure.
Output structure

The structure of the output:
- initial "egg" figure
- second "teacup" figure
- third "stop sign" figure
- fourth "hat" figure

This structure can be represented by functions:
- egg
- tea_cup
- stop_sign
- hat
def main():
    egg()
    tea_cup()
    stop_sign()
    hat()

def egg():
    print("  ______")
    print(" /\")
    print(" /\")
    print(" \ /")
    print(" \_____/")
    print()

def tea_cup():
    print(" \ /")
    print(" \_____/")
    print(" +----------+")
    print()

def stop_sign():
    print("  ______")
    print(" /\")
    print(" /\")
    print(" | STOP |")
    print(" \ /")
    print(" \_____/")
    print()

def hat():
    print("  ______")
    print(" /\")
    print(" /\")
    print(" +--------+")
    print()
Third version (structured, without redundancy):

- Identify redundancy in the output, and create functions to eliminate as much as possible.
- Add comments to the program.
Output redundancy

The redundancy in the output:

- egg top: reused on stop sign, hat
- egg bottom: reused on teacup, stop sign
- divider line: used on teacup, hat

This redundancy can be fixed by functions:

- egg_top
- egg_bottom
- line
# Suzy Student, CSc 110, Spring 2094
# Prints several figures, with methods for structure and redundancy.
def main():
    egg()
    tea_cup()
    stop_sign()
    hat()

# Draws the top half of an egg figure.
def egg_top():
    print("  ______")
    print("  /\
           \\
    
# Draws the bottom half of an egg figure.
def egg_bottom():
    print("  \\____/")
    print("  \ /   ")

# Draws a complete egg figure.
def egg():
    egg_top()
    egg_bottom()
    print()

# Draws a teacup figure.
def tea_cup():
    egg_bottom()
    line()
    print()

# Draws a stop sign figure.
def stop_sign():
    eggTop()
    print("| STOP |")
    egg_bottom()
    print()

# Draws a figure that looks sort of like a hat.
def hat():
    egg_top()
    line()

# Draws a line of dashes.
def line():
    print(" +--------+")