Announcements

- Reminders:
  - MPL Assignment 2 due 9/17 by 11:59pm
  - Program #2 due on 9/18 by 11:55pm
- Talks this week
  - Memphis Python User Group Talk
    • Tonight, 6:30pm, FJ-B
  - Careers Built on a Computing Degree Talk
    • Tomorrow, 4pm, FJ-B

Practice from Last Time

- You are in charge of desserts at Thanksgiving dinner. You decide to make 2 pumpkin pies and 1 apple pie.
- Write a program that defines these functions:
  - `make_apple()` should print a description of how to make an apple pie
  - `make_pumpkin()` should print a description of how to make a pumpkin pie
  - `main()` should call `make_apple()` and `make_pumpkin()` appropriately to make all the pies.
- Don’t forget to call `main()` at the end of your code!

Functions

- A function is a group of statements to which we assign a name.
  - Use the “def” keyword to define a function.
- That group of statements can then be referred to by that name later in the program.
  - Call a function by using its name with open/close parenthesis after it.
Function Example

```python
# This program has two functions. First we define the main function.
def main():
    print('I have a message for you.
    message()
    print('Goodbye!')

# Next we define the message function.
def message():
    print('I am Arthur')
    print('King of the Britons.')

# Call the main function.
main()
```

Output

```
I have a message for you.
I am Arthur
King of the Britons.
Goodbye!
```

Local Variables

- **Local variable**: variable that is assigned a value inside a function
  - Belongs to the function in which it was created
    - Only statements inside that function can access it, error will occur if another function tries to access the variable
  - **Scope**: the part of a program in which a variable may be accessed
    - For local variable: function in which created

- A local variable cannot be accessed by statements inside its function which precede its creation
- Different functions may have local variables with the same name
  - Each function does not see the other function’s local variables, so no confusion

Example

```python
# THIS PROGRAM DOESN'T WORK!

def sing_song():
    print("Happy birthday to you!
    Happy birthday to you!")
    print("Happy birthday dear", name,
    "Happy birthday to you!")

def main():
    name = input("What is your name? ")
    sing_song()

main()
```

name is a local variable – it is invisible to Python outside of the main function.

Attempting to use name here will cause an error.
Arguments and Parameters

**Defining:**
```python
def name_of_function(var1, var2, ...):
    statement
    statement
    statement
```

**Calling:**
```
nname_of_function(value1, value2, ..)
```

When a function is called, all the values inside the parentheses from the calling line are **immediately copied** into the variables given in the function definition.

---

**Defining:**
```python
def name_of_function(param1, param2, ...):
    statement
    statement
    statement
```

**Calling:**
```
nname_of_function(arg1, arg2, ..)
```

The values being copied from the calling function are called **arguments**. The variables being copied into are called **parameters**.

---

```python
def sing_song(name):
    print("Happy bday to you, happy bday to you!")
    print("Happy bday dear", name, "happy bday to you")

def main():
    my_name = input("What is your name? ")
    sing_song(my_name)
    twin_name = input("What is your twin's name? ")
    sing_song(twin_name)

main()
```

When Python runs the red line, it copies the value of my_name into sing_song's variable name.

```python
def sing_song(name):
    print("Happy bday to you, happy bday to you!")
    print("Happy bday dear", name, "happy bday to you")

def main():
    my_name = input("What is your name? ")
    sing_song(my_name)
    twin_name = input("What is your twin's name? ")
    sing_song(twin_name)

main()
```

When Python runs the blue line, it copies the value of twin_name into sing_song's variable name.
Recap

• There is no permanent connection between the \( x \) in `main` and the \( x \) in `some_function`.

• Arguments are passed \textit{one way only} from `main` to `some_function` when `main` calls `some_function`.
  – This copies `main`'s value of \( x \) into `some_function`'s \( x \).

• Any assignments to \( x \) inside of `some_function` do not come back to `main`.

Parameters = Local Variables

• “That sounds like local variables.”

• Just as local variables are invisible outside of the function that owns them, variables used as parameters inside a function definition are local to that function.

• Parameters in a function definition are really local variables that are created and assigned values automatically when the function is called.
You’ve seen arguments already.

- name = input("What is your name? ")
- x = 5
- y = 2
- print("x is", x, "y is", y)
- print("their sum is", x + y)

Arguments can be variables, literals, or math expressions.

In Class Example

- Using functions, write a program that prompts the user for 3 numbers and outputs the average of those numbers.

Tricky Example

```python
def mystery(x, z, y):
    print(z, y-x)

def main():
    x = 9
    y = 2
    z = 5
    mystery(z, y, x)
    mystery(y, x, z)
    mystery(x + z, y - x, y)

main()
```

Global Variables

- **Global variable**: created by assignment statement written **outside all the functions**
  - Can be accessed by any statement in the program file, including from within a function

- **DO NOT USE GLOBAL VARIABLES!**
  - Global variables making debugging difficult
    - Many locations in the code could be causing a wrong variable value
  - Functions that use global variables are usually dependent on those variables
    - Makes function hard to transfer to another program

Global variables make a program hard to understand!
Global Constants

- **Global constant**: global name that references a value that **cannot be changed**
  - OK to use global constants in a program
  - To simulate global constant in Python, create global variable and do not re-declare it within functions

Practice

- Write a Python program that asks the user to input 2 numbers and outputs the sum of those numbers.
  - Use 2 functions
    - `main()`: Prompts the user to enter 2 numbers and calls `sum()`
    - `sum()`: Takes in 2 parameters and outputs the sum of those numbers

Next Time

- Functions that return values
- Section 5.7-5.9