### **COMP 141**

Functions that take arguments, local variables



#### **Announcements**

- · Reminders:
  - Program #2 due on Thursday, February 1<sup>st</sup> by 11:55pm
  - Keep up with Zybooks assignments

2

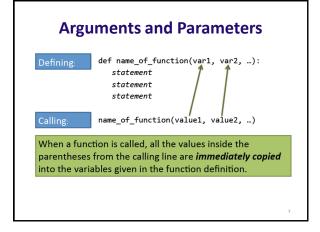
### **Practice from Last Time**

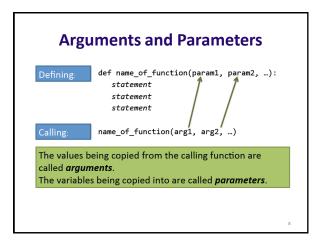
- You are in charge of desserts at Thanksgiving dinner.
   You decide to make 2 pumpkin pies and 1 apple pie.
- Write a program that defines these functions:
  - make\_apple() should print a description of how to make an apple pie
  - make\_pumpkin() should print a description of how to make a pumpkin 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
# This program has two functions. First we
# define the main function.
def main(): ←
   print('I have a message for you.')
                                            definitions
    message() 🖡
    print('Goodbye!')
# Next we define the message function.
                                            Function call
def message(): <
   print('I am Arthur')
   print('King of the Britons.')
# Call the main function.
                                      Output
main()
                              I have a message for you.
                               I am Arthur
                              King of the Britons.
```





```
def sing_song(name):
    print("Happy blay 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)
    When Python runs the red line, it copies the value of my_name into sing_song's variable name.
```

```
def sing_song(name):
    print("Happy blay 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.
```

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

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

    • You may use the same variable names in both places, if desired.
    • Each function then has its own copy of the variable.
    • There is no permanent link between the variables.
```

```
def some_function(x):
    print("Inside the function, x is", x)
    x = 17
    print("Inside the function, x is changed to", x)

def main():
    x = 2
    print("Before the function call, x is", x)
    some_function(x)
    print("After the function call, x is", x)

    Output:
main()
    Before the function call, x is 2
    Inside the function, x is 2
    Inside the function, x is 17
    After the function call, x is 2
```

### Recap

- There is no permanent connection between the  $\mathbf x$  in main and the  $\mathbf x$  in some function.
- Arguments are passed ---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.

13

#### **Local Variables**

- <u>Local variable</u>: 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

14

### **Local Variables**

- 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

# 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.

18

# **Tricky Example**

```
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**

- <u>Global variable</u>: 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

# # The following is used as a global constant to represent Global # the contribution rate. CONTRIBUTION RATE = 0.05 def main(): gross\_pay = float(input('Enter the gross pay: ')) bonus = float(input('Enter the amount of bonuses: ')) show pay\_contrib(gross\_pay) show\_bonus\_contrib(bonus) **Constant Example** # Call the main function. main()

### **Practice**

### 1. Modify singHappyBirthday.py

- You no longer have a twin. Now you have a sibling that is two years older than you, but you share the same birthday.
- Edit code so that sing\_song now will print the lyrics but also print how old the person is.
- Add a second parameter to sing\_song called age.
- Edit main() to ask for your age, as well as your name and sibling's name.
- Edit the two calls to sing\_song so appropriate ages are passed as arguments.
- 2. Write a new 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