

# CS 115, Autumn 2021

## Lecture 26: returns



*"There's your problem...  
You've got an extra parenthesis in line 18."*

Thanks to Marty Stepp and Stuart Reges for parts of these slides

# Getting data from functions

- The following code prints the biggest number successfully
  - What if we wanted to use this number later on in our program?
    - Can we print `highest` in `main` or use it in a computation?

```
def main():
    num1 = int(input("First? "))
    num2 = int(input("Second? "))
    num3 = int(input("Third? "))
    biggest(num1, num2, num3)

def biggest(num1, num2, num3):
    highest = num1
    if num2 > num1 and num2 > num3:
        highest = num2
    elif num3 > num1 and num3 > num2:
        highest = num3
    print("The biggest number is", highest)

main()
```

# Getting data from functions

- The following code prints the biggest number successfully
  - What if we wanted to use this number later on in our program?
    - Can we print `highest` in main or use it in a computation?

```
def main():
    num1 = int(input("First? "))
    num2 = int(input("Second? "))
    num3 = int(input("Third? "))
    biggest(num1, num2, num3)

def biggest(num1, num2, num3):
    highest = num1
    if num2 > num1 and num2 > num3:
        highest = num2
    elif num3 > num1 and num3 > num2:
        highest = num3
    print("The biggest number is", highest)

main()
```

# Problem: scope

- Variables only exist in the function they are created in
  - `highest` doesn't exist in `main`!

```
def main():  
    num1 = int(input("First? "))  
    num2 = int(input("Second? "))  
    num3 = int(input("Third? "))  
    biggest(num1, num2, num3)  
    print("The biggest number is", highest)
```

```
def biggest(num1, num2, num3):  
    highest = num1  
    if num2 > num1 and num2 > num3:  
        highest = num2  
    elif num3 > num1 and num3 > num2:  
        highest = num3  
    print("The biggest number is", highest)
```

```
main()
```

# Scope

- How have we fixed scope problems before?
  - Parameters!
  - Can we use parameters to fix this?

```
def biggest(num1, num2, num3, highest):  
    highest = num1  
    if num2 > num1 and num2 > num3:  
        highest = num2  
    elif num3 > num1 and num3 > num2:  
        highest = num3
```

This wont work!

# Value semantics

- **value semantics:** When value types (`int`, `double`) are passed as parameters, their values are copied.
  - Modifying the parameter will not affect the variable passed in.

```
def main():  
    x = 23  
    strange(x)  
    print("2. x =", x)
```

```
def strange(x):  
    x = x + 1  
    print("1. x =", x)
```

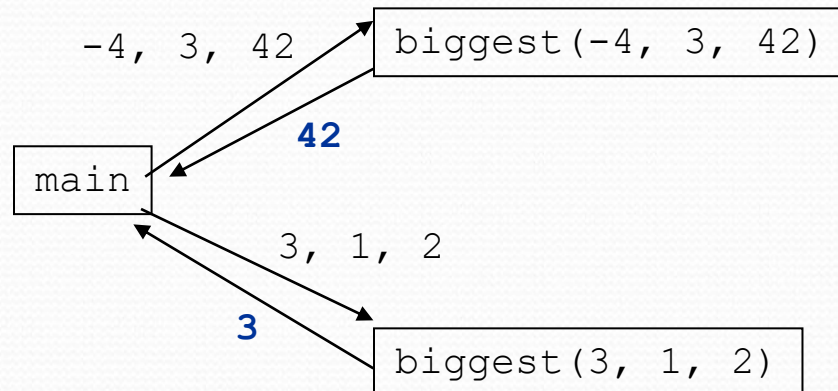
```
main()
```

Output:

```
1. x = 24  
2. x = 23
```

# Solution: `return`

- **return:** To send out a value as the result of a function.
  - Return values send information *out* from a function to its caller.
  - A call to the function can be used as part of an expression.
  - (Compare to parameters which send values *into* a function)



# Returning a value

```
def name (parameters) :  
    statements  
    ...  
    return expression
```

- When Python reaches a return statement:
  - it evaluates the expression
  - it substitutes the return value in place of the call
  - it goes back to the caller and continues after the function call

# Return examples

```
# Converts degrees Fahrenheit to Celsius.
```

```
def f_to_c(degrees_f):  
    degrees_c = 5.0 / 9.0 * (degrees_f - 32)  
    return degrees_c
```

```
# Computes triangle hypotenuse length given its side lengths.
```

```
def hypotenuse(a, b):  
    c = (a * a + b * b)**(1/2)  
    return c
```

- You can shorten the examples by returning an expression:

```
def f_to_c(degrees_f):  
    return 5.0 / 9.0 * (degrees_f - 32)
```

# Common error: Not storing

- Many students incorrectly think that a `return` statement sends a variable's name back to the calling function.

```
def main():
    slope(0, 0, 6, 3)
    print("The slope is", result); # ERROR: cannot find
                                  # symbol: result

def slope(x1, x2, y1, y2):
    dy = y2 - y1
    dx = x2 - x1
    result = dy / dx
    return result

main()
```

# Fixing the common error

- Returning sends the variable's *value* back. Store the returned value into a variable or use it in an expression.

```
def main():  
    s = slope(0, 0, 6, 3)  
    print("The slope is", s)
```

```
def slope(x1, x2, y1, y2):  
    dy = y2 - y1  
    dx = x2 - x1  
    result = dy / dx  
    return result
```

```
main()
```

# Fixing the common error

- We have actually used functions that return values before:
  - `randint`  

```
number = randint(1, 10)
```
  - `input`  

```
answer = input("What is the answer? ")
```
  - `drawing_panel`  

```
panel = drawing_panel(300, 300)
```
- Notice that whenever we have called these functions, we have stored the result in a variable

# Exercise

- In physics, the *displacement* of a moving body represents its change in position over time while accelerating.
  - Given initial velocity  $v_0$  in m/s, acceleration  $a$  in  $\text{m/s}^2$ , and elapsed time  $t$  in s, the displacement of the body is:
    - Displacement =  $v_0 t + \frac{1}{2} a t^2$
- Write a function `displacement` that accepts  $v_0$ ,  $a$ , and  $t$  and computes and returns the change in position.
  - example: `displacement(3.0, 4.0, 5.0)` returns 65.0

# Exercise solution

```
def displacement(v0, t, a):  
    d = v0 * t + 0.5 * a * (t ** 2)  
    return d
```

# Exercise

- If you drop two balls, which will hit the ground first?
  - Ball 1: height of 600m, initial velocity = 25 m/sec downward
  - Ball 2: height of 500m, initial velocity = 15 m/sec downward
- Write a program that determines how long each ball takes to hit the ground (and draws each ball falling).
- Total time is based on the force of gravity on each ball.
  - Acceleration due to gravity  $\cong 9.81 \text{ m/s}^2$ , downward
  - Displacement =  $v_0 t + \frac{1}{2} a t^2$

# Ball solution

```
# Simulates the dropping of two balls from various heights.
```

```
def main():  
    panel = drawing_panel(600, 600)  
    ball1x = 100  
    ball1y = 0  
    v01 = 25  
    ball2x = 200  
    ball2y = 100  
    v02 = 15  
  
    # draw the balls at each time increment  
    for time in range(60):  
        disp1 = displacement(v01, time/10, 9.81)  
        panel.fill_oval(ball1x, ball1y + disp1, ball1x + 10, ball1y + 10 + disp1)  
        disp2 = displacement(v02, time/10, 9.81)  
        panel.fill_oval(ball2x, ball2y + disp2, ball2x + 10, ball2y + 10 + disp2)  
  
        panel.sleep(50)    # pause for 50 ms  
        panel.clear()  
  
def displacement(v0, t, a):  
    d = v0 * t + 0.5 * a * (t ** 2)  
    return d  
  
main()
```