

# Functions that return a value

## Lecture 08

Professor Alan Broder

[alan.broder@yu.edu](mailto:alan.broder@yu.edu)

Some material with permission from Stuart Reges & Marty Stepp

## Examples of functions that we've already encountered that return a value

Example usage	Explanation
<code>s = str(9*2)</code>	Takes a value as input and returns a string rendering of that value.
<code>age = input("enter your age:")</code>	Takes as input a string to prompt the user with, and returns a string that the user typed in at the keyboard.
<code>years = int(age)</code>	Takes a value as input (e.g. string, numeric value), and returns the integer representation of that input
<code>Brown = Draw.color(192,128,64)</code>	Takes R, G, B intensities as input and returns a Color object that can be used later in <code>Draw.setColor()</code>

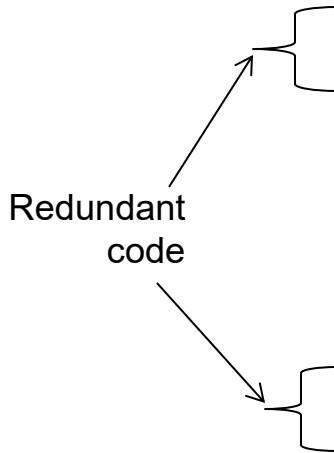
# Computing the sum of integers

- Suppose we want to compute the sum of the integers between `low` and `high` :

Redundant code

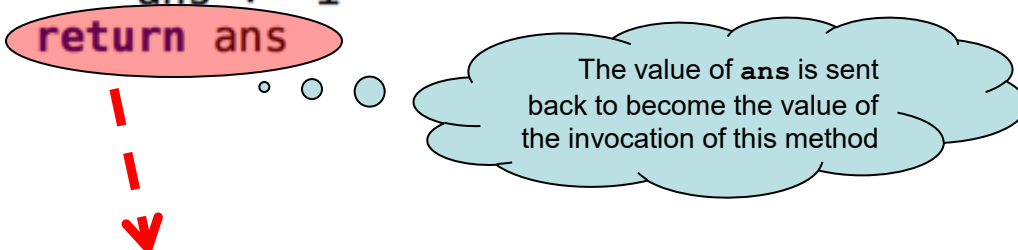
```
sum = 0
low = 2
high = 8
for i in range(low, high+1):
    sum += i
print("The sum is " + str(sum))

sum = 0
low = 12
high = 34
for i in range(low, high+1):
    sum += i
print("The sum is " + str(sum))
```



# Defining a function that returns a value

```
def sumRange(low, high):  
    ans = 0  
    for i in range(low, high+1):  
        ans += i  
    return ans
```



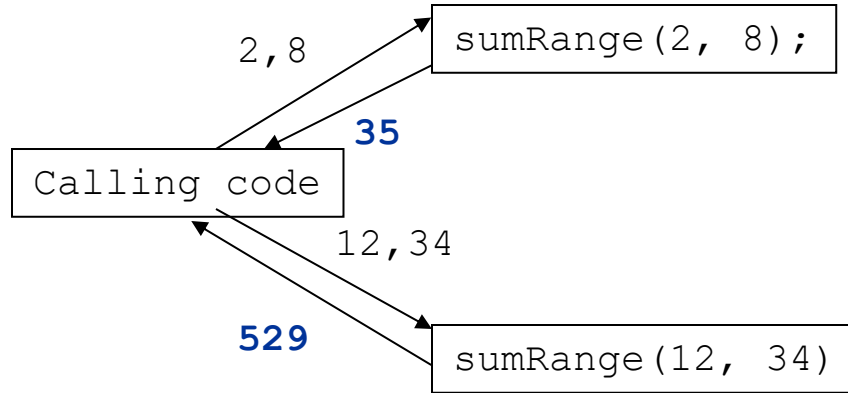
The value of `ans` is sent back to become the value of the invocation of this method

```
sum = sumRange(2, 8)  
print("The sum is: " + str(sum))
```

```
sum = sumRange(12, 34)  
print("The sum is: " + str(sum))
```

# The return statement

- **return:** To send back a value as the result of a function.
  - The opposite of a parameter:
    - Parameters send information from the caller **into** the function.
    - Return values send information from a function back **out** to its caller.
  - A call to a function can be used as part of an expression.



# Defining a function that returns a value

```
def sumRange(low, high):  
    ans = 0  
    for i in range(low, high+1):  
        ans += i  
    return ans  
  
sum = sumRange(2, 8)  
print("The sum is: " + str(sum))  
  
sum = sumRange(12, 34)  
print("The sum is: " + str(sum))
```

The diagram illustrates the execution of the `sumRange` function. A blue dashed arrow points from the `sumRange(2, 8)` call to the `return ans` statement in the function definition. Two red dashed arrows point from the `sumRange(12, 34)` call to the `return ans` statement in the function definition. Additionally, a red dashed arrow points from the `return ans` statement back to the `sum` variable in the `sum = sumRange(2, 8)` call, indicating the return value being assigned.

# Using the return statement

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

- Example:

```
# define the function  
def slope(x1, y1, x2, y2):  
    dy = y2 - y1  
    dx = x2 - x1  
    return dy / dx  
  
# now invoke the function  
s = slope(1, 3, 5, 11) # returns 2.0  
print("The slope is: " + str(s))  
  
s = slope(2, 4, 14, 15) # returns 0.91666666  
print("The slope is: " + str(s))
```

# if/else with return

# Returns the larger of the two given numbers.

```
def larger(a, b):  
    if a > b:  
        return a  
    else:  
        return b
```

- Functions can return different values using `if/elif/else`
  - Whichever path the code enters, it will return that value.
  - Returning a value causes a function to immediately exit.
  - All paths through the function must reach a `return` statement
  - **If you forget to provide a return, and execution of the function “falls off the bottom”, the function will return a value called `None`**



# Poll Time! #1

What does this complete program print when run?

```
def checker(x, y):  
    if x < 1:  
        return y  
    elif y < x:  
        return x + y + 1  
  
def main():  
    ans = checker(2, 9)  
  
main()
```

## Sample of already-available built-in functions

- `min(a,b,c,...)` – returns the smallest of the input values
- `max(a,b,c...)` – returns the largest of the input values
- `abs(x)` – returns the absolute value of the input value
- `float(x)` – returns the floating point version of the input expression or string
- `int(x)` – returns the integer version of the input expression or string
- `str(x)` – returns the string representation of the input expression or numeric value
- `round(x)` – returns the nearest whole number (as an int) to the input expression

**MANY, MANY additional “modules” of functions are available.**  
**For example, a small sample of functions in the `math` module:**

<b>Function declaration:</b>	<b>Returns (unless noted, all return a float value)</b>
<code>math.ceil(a)</code>	<b>a</b> rounded up to the nearest whole number – returns an int
<code>math.floor(a)</code>	<b>a</b> rounded down to the nearest whole number – returns an int
<code>math.log10(a)</code>	Logarithm base 10 of <b>a</b>
<code>math.pow(a, b)</code>	<b>a</b> raised to the <b>b</b> 'th power (you can also use <code>**</code> )
<code>math.sqrt(a)</code>	Square root of <b>a</b> (you can also use <code>**</code> )
<code>math.cos(a)</code>	Cosine of <b>a</b> (where <b>a</b> is expressed in radians)
<code>math.sin(a)</code>	Sine of <b>a</b> (where <b>a</b> is expressed in radians)
<code>math.tan(a)</code>	Tangent of <b>a</b> (where <b>a</b> is expressed in radians)
<code>math.degrees(a)</code>	Convert angle <b>a</b> from radians to degrees
<code>math.radians(a)</code>	Convert angle <b>a</b> from degrees to radians

<b>Math constants</b>	<b>Description</b>
<code>math.e</code>	2.7182818...
<code>math.pi</code>	3.1415926...

# Calling math functions

- Examples:



Must have this

```
import math # import math needed just once
```

```
squareRoot = math.sqrt(121.0)
print(squareRoot) # 11.0
```

```
val = math.sin(math.radians(90))
print(val) # 1.0
```

- The `math` functions ***do not print*** to the console.
  - Each functions produces (i.e. returns) a numeric result.
  - The results are used in expressions (printed, stored, etc.).
- **Many** more math and other modules of functions can be found at:
  - <https://docs.python.org/3/py-modindex.html>
  - <https://pypi.org/>

# Math and built-in function questions

- Evaluate the following expressions:

- `abs(-1.23)`
- `math.sqrt(121.0) - math.sqrt(256.0)`
- `math.ceil(6.022) + math.floor(15.9994)`
- `abs(min(-3, -5))`
- `math.pow(2.0, 3)`

- `max` and `min` can be used to put a bound on numbers.

Consider a variable named `age`.

- What statement would replace negative ages with 0?
- What statement would cap the maximum age to 40?

# Review: type casting

- **type cast:** A conversion from one type to another.
  - To promote an `int` into a `float`
  - To truncate a `float` from a real number to an integer
- Syntax:

**type (expression)**

Examples:

```
result = float(10+9)           # 19.0
result2 = int(result)          # 19
x = int(math.pow(10, 3))       # 1000
```

- A conversion to `float` can be achieved in other ways.

```
average = 1.0 * (a + b + c) * 3
```

## return example using math module

```
import math
```

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

```
def fToC(degreesF):  
    degreesC = 5.0 / 9.0 * (degreesF - 32)  
    return degreesC
```

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

```
def hypotenuse(a, b):  
    c = math.sqrt(a * a + b * b)  
    return c
```

- You can shorten the examples by returning the value of an expression:

```
def fToC(degreesF):  
    return 5.0 / 9.0 * (degreesF - 32)
```

```
def hypotenuse(a, b):  
    return math.sqrt(a * a + b * b)
```

# Poll Time! #2

What does this code print?

```
def adder(a, b):  
    return a + b
```

```
def main():  
    print("The sum is " + adder(3, 4))
```

```
main()
```



## DON'T MAKE THIS CONCEPTUAL ERROR !

- Some students mistakenly conclude that the **return** statement somehow causes a value to be printed by their program.
- This is not true!
  - Only `print` can cause something to be printed.
  - **If you don't see a `print` statement then nothing will be printed by your program!**

# Common error: Not storing

- Some students incorrectly think that a `return` statement sends a variable's ***name*** back to the calling code.

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

```
def main():  
    slope(10, 0, 6, 3)  
    print("The slope is" + str(result))  
    # ERROR name 'result' is not defined  
main()
```

# Fixing the common error

- Instead, `return` sends back a reference to the variable's ***value***.
  - The returned value must be stored into a variable or used in an expression to be useful to the caller.

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

```
def main():  
    s = slope(9, 0, 6, 3)  
    print("The slope is " + str(s))          # 0.3333333333333333  
main()
```

# Getting a random number

- Use the `random.random()` function from the `random` module
- Example:

```
import random
```

```
# print a single random number in [0,1)  
print(random.random())
```

```
# print 10 random numbers in [0,1)  
for i in range(10):  
    print(random.random())
```

# Examples

- Write a function that takes four input parameters a, b, c, and d and **returns** the larger of the sum of a and b, or c and d
- Write a function that **returns** a random throw of a die (i.e. a number between 1 and 6, inclusive)
- Write a function that **returns** a random flip of a coin (i.e. randomly returns the string "heads" or "tails")

# Semester Project Proposal

## Due at the latest by November 5

- Upload outline of proposed semester project to **Canvas** dropbox before class November 5
- Describe:
  - What will your program do? (describe in detail, including any interaction with a user)
  - What type of graphics do you plan to use (if any): Draw.py graphics? text “graphics” ?
  - If you plan to use any external data source – describe it.
- Very broad guidelines for the actual project:
  - Your program should result in an **absolute minimum** of 100 lines of Python code (not including blank lines, comment lines, or supplemental code that I furnish).
    - However, merely satisfying the 100-line minimum does not constitute on its own a satisfactory submission. ***The code has to do what you proposed, even if (many) more lines of code are needed.***
  - Must be **your own code** – make sure you understand the consequences of the Syllabus policy regarding academic integrity
  - If you are choosing the “optical illusion” type of program, you must create at least two illusions
  - Points are deducted for:
    - Inadequately commented code
    - Code with redundancy that could have been fixed with functions or methods
    - Code with gratuitous global variables
    - Code that crashes (I will test interactive programs and try to make them crash)
    - Code that doesn’t do what your proposal described (i.e. bugs or missing features)

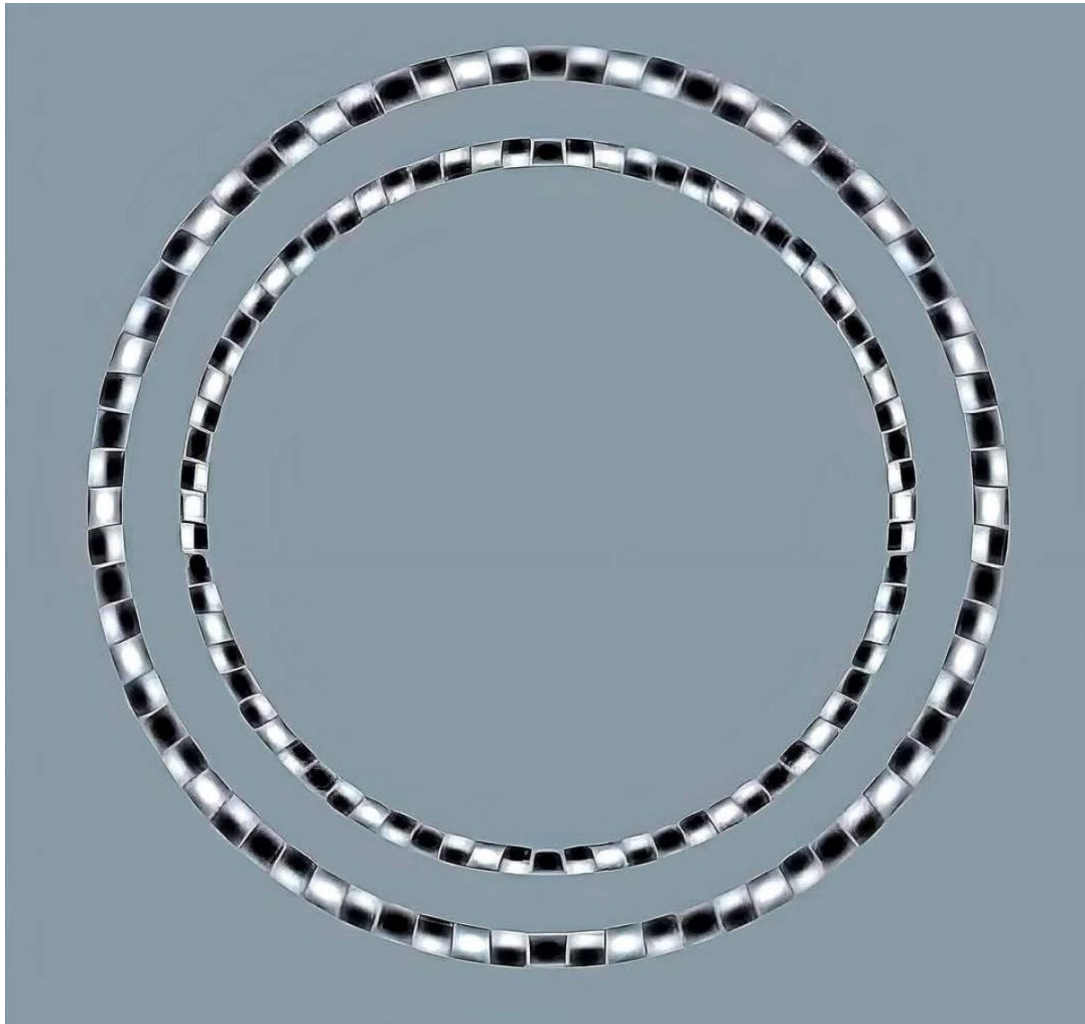
# For inspiration: Demo Reels of Prior Year Semester Projects

- 2015-2016: <https://youtu.be/iXfGZdTZ32k>
- 2017: <https://youtu.be/HzOR9IS3osE>
- 2018: <https://youtu.be/l7-TOTkq28g>
- 2019: <https://youtu.be/0aGpnDbtNQ0>
- 2020: <https://drive.google.com/file/d/1EZWRcC6osUfrqpHgZts7jSDz1BNG3Ue9/view>

# Examples of semester projects done in prior semesters

- Science: DNA visualization, solar system simulation, titration demo
- Othello, Sudoku, Wordle, Jeopardy, 2048
- Optical illusions (must do minimum of 2, depending on degree of difficulty)
  - VERY OVERDONE – SO YOU WILL NEED TO PROPOSE SOMETHING VERY SPECIAL
  - BEWARE THAT ILLUSION PROJECTS WILL NOT LOOK GOOD ON YOUR TECHNICAL RESUME!
  - see examples (warning – may induce dizziness/nausea!) at <http://www.ritsumei.ac.jp/~akitaoka/index-e.html>
  - Students have done: autumn color swamp, bulge, primrose field, rollers, “the music” (page 0)
  - Feel free to browse the site for other illusions, but you’ll want to select illusions that can be done with the graphics methods we’ve learned (many can). Avoid illusions that use smooth shading (as opposed to shapes of solid colors).
- Play against the computer games: Battleship, Go Fish (play against the computer), blackjack
- More games: Tetris, Pac Man, Candy Crush, Frogger, Snake, Set, Asteroids, Hangman
- Overdone programs (not likely to be approved):
  - Connect 4, Mastermind, Madlibs, Simple Optical Illusions





# Project Proposal Process

- Choose a “pre-approved” project from the following list, all of which must use Draw.py:
  - Wordle – must use a list of many possible words
  - Snake – must use input from the keyboard to direct the snake
  - Memory game – must use GIF pictures that you furnish
  - Hangman – must use a file of many possible words
- **OR**, Propose your own one or two ideas, and I will let you know which one I approve
- Whichever approach you choose, this must be **entirely** your own work. Absolutely no collaboration allowed.

# Lab

## •Lab –

- There is a Google form in the Canvas Lab 08 assignment
- Complete the form and submit it.
- Remember that:
  - `math.log10()` returns the logarithm of the input value to the base 10.
  - `round()` returns the input value rounded up or down to the nearest whole number. If the input value is exactly in the middle of the two whole numbers, the answer is rounded up (e.g. 2.5 rounds to 3).
  - `random.random()` returns a random number in the range [0, 1)

## Homework due before next class

- **Homework** – Use Wing to solve the three problems on the next few pages. Upload to the HW08 assignment at [codePost.io](https://codePost.io) three files containing your Python code.

# Homework Problem #1

## `lastdigit.py`

- Using Wing, write and test a function called `lastDigit` that returns the last digit of an integer. For example, `lastDigit(3572)` should return 2. ***It must work for negative numbers as well.*** For example, `lastDigit(-947)` should return 7.
- Test the function in your Python file by writing a `main` function that calls `lastDigit` several times with different parameters, and using `print` to print out the returned value.
  - **In order to get full credit for this problem your submitted code must include a `main()` function with these tests**
- Hint: don't forget about the `%` (mod) operator !

## Homework Problem #2

`circlearea.py`

- Using Wing, write and test a function called `circleArea` that accepts the radius of a circle as a parameter, and returns the area of a circle with that radius. For example, the call `circleArea(2)` should return 12.5663706144. You may assume that the radius passed to `circleArea` is always non-negative.
- Test the function in your Python file by writing a `main` function that calls `circleArea` several times with different parameters, and using `print` to print out the returned value.
  - **In order to get full credit for this problem your submitted code must include a main function with these tests**

# Homework Problem #3

## trianglearea.py

- Using Wing, write and test a function called `triangleArea` that accepts the three side lengths of a triangle as parameters and returns the area of a triangle with those side lengths.
- For example, the call `triangleArea(8, 5, 6)` should return 14.9812382666, and `triangleArea(1, 2, 2.5)` should return 0.9499177595981665, and `triangleArea(1, 1, 1)` should return 0.4330127018922193. To compute the area, use Heron's formula, which states that the area of a triangle whose three sides have lengths  $a$ ,  $b$ , and  $c$ , is the following. The formula is based on the computed value of  $s$ , a length equal to half the perimeter of the triangle.

$$s = (a + b + c) / 2$$

where

$$area = \sqrt{s(s-a)(s-b)(s-c)}$$

- Test the function in your Python file by writing a `main` function that calls `triangleArea` several times with different parameters, and using `print` to print out the returned value. You should validate your results by doing the calculation by hand using a calculator.
  - **In order to get full credit for this problem your submitted code must include a main function that does these tests**