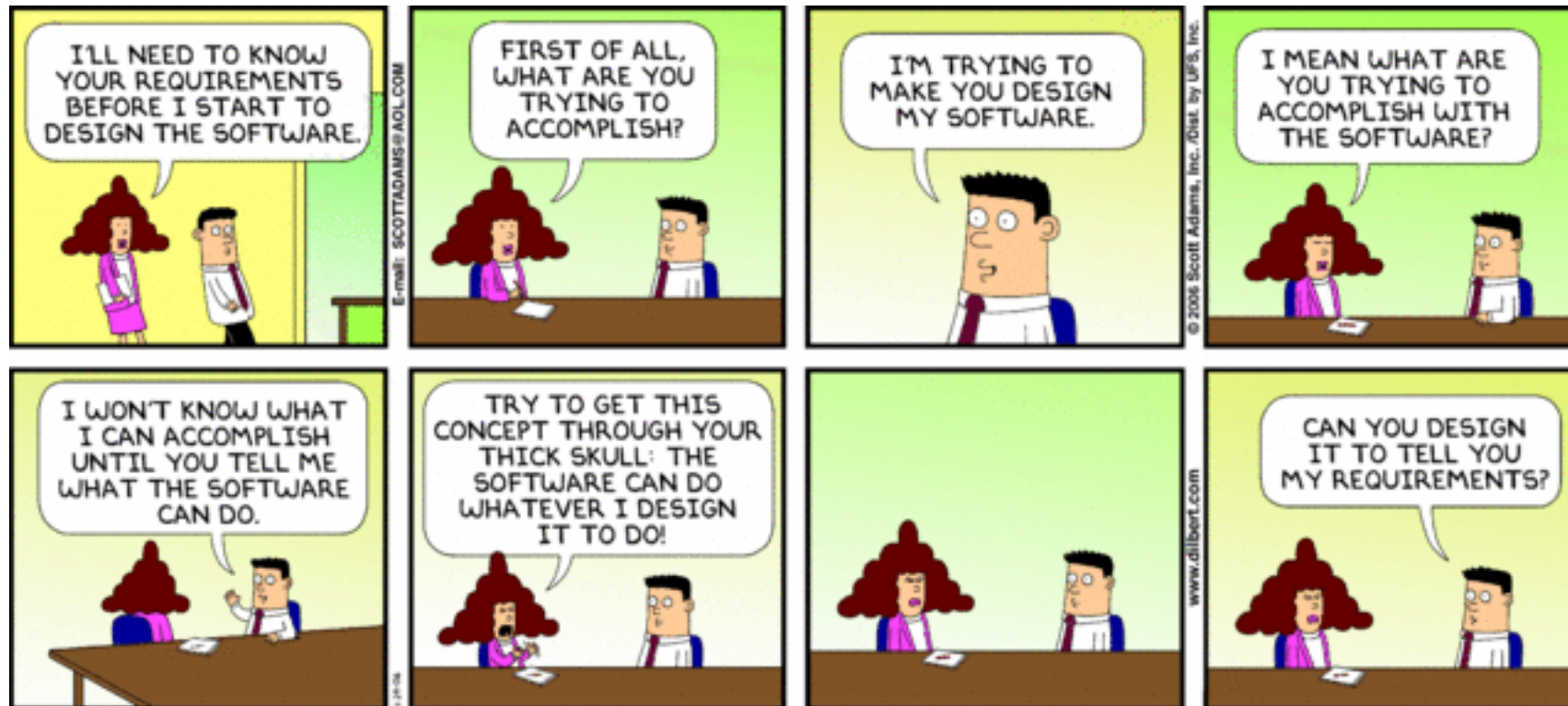


# CSc 110, Spring 2018

## Lecture 11: return values and `math`



# Python's Math class

Method name	Description
<code>math.ceil(<i>value</i>)</code>	rounds up
<code>math.floor(<i>value</i>)</code>	rounds down
<code>math.log(<i>value</i>, <i>base</i>)</code>	logarithm
<code>math.sqrt(<i>value</i>)</code>	square root
<code>math.sinh(<i>value</i>)</code> <code>math.cosh(<i>value</i>)</code> <code>math.tanh(<i>value</i>)</code>	sine/cosine/tangent of an angle in radians
<code>math.degrees(<i>value</i>)</code> <code>math.radians(<i>value</i>)</code>	convert degrees to radians and back

Constant	Description
<code>e</code>	2.7182818...
<code>pi</code>	3.1415926...

`import math`      necessary to use the above functions

## Other math functions:

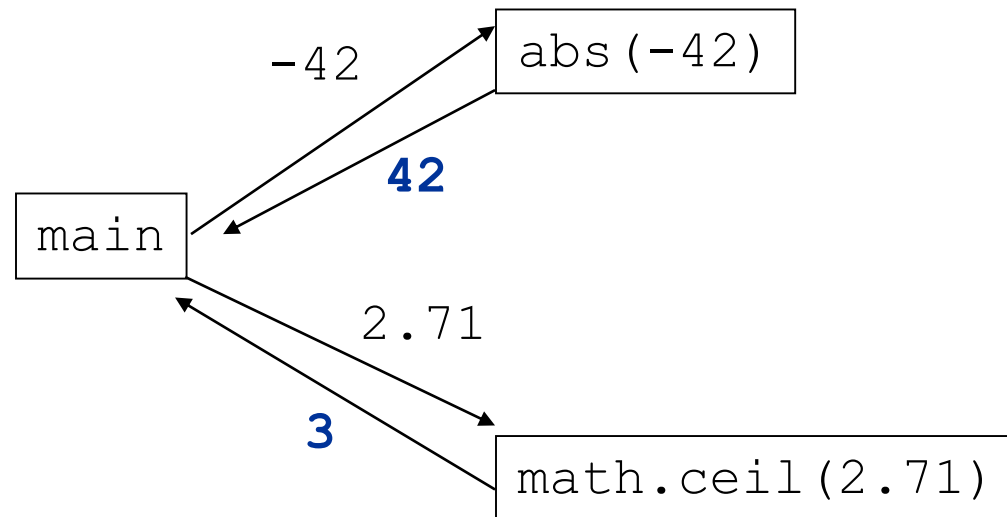
Function name	Description
<code>abs(<i>value</i>)</code>	absolute value
<code>min(<i>value1</i>, <i>value2</i>)</code>	smaller of two values
<code>max(<i>value1</i>, <i>value2</i>)</code>	larger of two values
<code>round(<i>value</i>)</code>	nearest whole number

# No output?

- Simply calling these functions produces no visible result.
  - `math.sqrt(81)`      `# no output`
- Math function calls use a Python feature called *return values* that cause them to be treated as expressions.
- The program runs the function, computes the answer, and then "replaces" the call with its computed result value.
  - `math.sqrt(81)`      `# no output`  
`9.0`      `# no output`
- To see the result, we must print it or store it in a variable.
  - `result = math.sqrt(81)`
  - `print(result)`      `# 9.0`

# 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)



# Math questions

- Evaluate the following expressions:
  - `abs(-1.23)`
  - `math.sqrt(121.0) - math.sqrt(256.0)`
  - `round(pi) + round(e)`
  - `math.ceil(6.022) + math.floor(15.9994)`
  - `abs(min(-3, -5))`
- `math.max` and `math.min` can be used to bound numbers.  
Consider a variable named `age`.
  - What statement would replace negative ages with 0?
  - What statement would cap the maximum age to 40?

# Quirks of real numbers

- Some `float` values print poorly (too many digits).

```
result = 1.0 / 3.0  
print(result)          # 0.3333333333333333
```

- The computer represents `floats` in an imprecise way.

```
print(0.1 + 0.2)
```

- Instead of 0.3, the output is `0.30000000000000004`

# Type casting

- **type cast:** A conversion from one type to another.
  - To truncate a `double` from a real number to an integer

- Syntax:

**type (expression)**

Examples:

```
result = 19 / 5
```

```
# 3.8
```

```
result2 = int(result)
```

```
# 3
```

```
x = int(sqrt(121))
```

```
# 1000
```

# 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 method 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 = math.sqrt(a * a + b * b)  
    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 method.

```
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
```

# 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
```

# 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 method `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, a, t):  
    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 = DrawingPanel(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.fill_rect(0, 0, 600, 600, "white")
```

```
...
```