CSc 110, Autumn 2016

Lecture 8: Return values and math
Modify `draw_car` to allow the car to be drawn at any size.

- Existing car: size 100. Second car: (150, 10), size 50.

Once you have this working, use a `for` loop with your function to draw a line of cars, like the picture at right.

- Start at (10, 130), each size 40, separated by 50px.
def main():
    panel = DrawingPanel(260, 100, background="light gray")
    draw_car(panel, 10, 30, 100)
    draw_car(panel, 150, 10, 50)
    for i in range(0, 5):
        draw_car(panel, 10 + i * 50, 130, 40);

def draw_car(p, x, y, size):
    p.canvas.create_rectangle(x, y, x + size, y + size / 2, fill="black")
    p.canvas.create_oval(x + size / 10, y + size / 10 * 4, x + size / 10 * 3, y + size / 10 * 6, fill="red", width=0)
    p.canvas.create_oval(x + size / 10 * 7, y + size / 10 * 4, x + size / 10 * 9, y + size / 10 * 6, fill="red", width=0)
    p.canvas.create_rectangle(x + size / 10 * 7, y + size / 10, x + size, y + size / 10 * 3, fill="cyan", width=0)
Python's Math class

<table>
<thead>
<tr>
<th>Method name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ceil(value)</td>
<td>rounds up</td>
</tr>
<tr>
<td>floor(value)</td>
<td>rounds down</td>
</tr>
<tr>
<td>log(value, base)</td>
<td>logarithm</td>
</tr>
<tr>
<td>sqrt(value)</td>
<td>square root</td>
</tr>
<tr>
<td>sinh(value)</td>
<td>sine/cosine/tangent of an angle in radians</td>
</tr>
<tr>
<td>cosh(value)</td>
<td></td>
</tr>
<tr>
<td>tanh(value)</td>
<td></td>
</tr>
<tr>
<td>degrees(value)</td>
<td>convert degrees to radians and back</td>
</tr>
<tr>
<td>radians(value)</td>
<td></td>
</tr>
</tbody>
</table>

Other math functions:

<table>
<thead>
<tr>
<th>Function name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>abs(value)</td>
<td>absolute value</td>
</tr>
<tr>
<td>min(value1, value2)</td>
<td>smaller of two values</td>
</tr>
<tr>
<td>max(value1, value2)</td>
<td>larger of two values</td>
</tr>
<tr>
<td>round(value)</td>
<td>nearest whole number</td>
</tr>
</tbody>
</table>
No output?

• Simply calling these functions produces no visible result.
  • \( \sqrt{81} \) # no output

• Math function calls use a Python feature called \textit{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.
  • \( \sqrt{81} \) # no output
  9.0 # no output

• To see the result, we must print it or store it in a variable.
  • \texttt{result = \sqrt{81}}
  • \texttt{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:
  • $\text{abs}(-1.23)$
  • $\sqrt{121.0} - \sqrt{256.0}$
  • $\text{round}(\pi) + \text{round}(e)$
  • $\text{ceil}(6.022) + \text{floor}(15.9994)$
  • $\text{abs}($min$(-3, -5))$

• max and 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).
  ```python
  result = 1.0 / 3.0
  print(result)  # 0.3333333333333333
  ```

• The computer represents `floats` in an imprecise way.
  ```python
  print(0.1 + 0.2)
  ```
  • Instead of 0.3, the output is `0.3000000000000004`
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 = 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.

```python
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.

```python
def main():
    s = slope(0, 0, 6, 3)
    print("The slope is " + str(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 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
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 \( \approx 9.81 \text{ m/s}^2 \), 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(0, 60, 1):
        disp1 = displacement(v01, time/10, 9.81)
        panel.canvas.create_oval(ball1x, ball1y + disp1, ball1x + 10, ball1y + 10 + disp1)
        disp2 = displacement(v02, time/10, 9.81)
        panel.canvas.create_oval(ball2x, ball2y + disp2, ball2x + 10, ball2y + 10 + disp2)

        panel.sleep(50)  # pause for 50 ms
        panel.canvas.create_rectangle(0, 0, 600, 600, fill="white", width=0)

...