Introduction to: Computers & Programming: Loops in Python: Examples

Adam Meyers
New York University
Printing a Multiplication table

• def multiplication_table (high_num):
  for num1 in range(1, 1+high_num):
    for num2 in range(1, 1+high_num):
      print(num1,'X',num2, '=' , num1*num2)

• How does this work?
Example: Drawing an asterisk triangle

• def draw_n_asterisks(n):
  
  for current_length in range(n):
    print('*',end='')
  
  – print can take a named argument
  • End='' indicates what to print at the end of the string
    – the character in between the single quotes
  • In this case, nothing
  • The default is a newline character

• def asterisk_triangle(base_size):
  
  for current_length in range(1,base_size+1):
    draw_n_asterisks(current_length)
  
  print()
Drawing an asterisk triangle 2

• Nested Loops – a single function

```python
def asterisk_triangle2(base_size):
    for current_length in range(1, base_size + 1):
        for n in range(current_length):
            print('*', end='')
        print()
```

• Python indicates depth of nesting via indentation
  – Suppose the last line was indented once
Sample Problem for Class

• Write a function that:
  – Takes three arguments:
    • base_size
    • repetitions
    • hour_glass_or_diamond
  – This function makes a pattern of asterisks that repeats the number of times indicated by repetitions
  – Each cycle consists of two triangles, one the upside down version of each other, both of which have a base of size base_size
  – If hour_glass_or_diamond is in the 'hour glass' setting, the function draws an upside down triangle and then a right side up triangle
  – If hour_glass_or_diamond is in the 'diamond' setting, the function draws the right side up triangle first and the upside down one second
One Way to Describe a Checkerboard

- A Checkerboard is an 8 X 8 square with alternating colors, e.g., red and black.
- A Checkerboard can be broken down into 4 bars, each a 2 X 8 bar of alternating colors.
- A 2 X 8 bar of alternating colors can be broken down into 4 composite squares, each consisting of 2 X 2 small squares.
A Pictoral Description

\[ \times 4 = \quad \times 4 = \]

\[ \times 4 = \]

Intro to: Computers & Programming:
Loops in Python
CSCI-UA.0002
One Way A Turtle Can Draw a Checkerboard?

- The turtle can draw one square and it could fill in with a color of our choice
- The turtle can make four such squares next to each other, forming a composite square.
- It can make four composite squares next to each other, to form a bar.
- It can make four such bars, one under the other to form a checkerboard.
for_loop_checkerboard.py 1

• Basic setup
  – import turtle
  – my_screen = turtle.Screen()
  – my_screen.setup(0.5,0.75,0,0)
    • width, height, startx, starty
  – my_turtle = turtle.Turtle(shape='turtle')

• draw_colored_turtle_square
• draw_4_black_and_red
• draw_4_black_and_red_4_times
• make_checkerboard
for_loop_checkerboard.py

• Setup and then do something 4 times
  – Building block
  – Move in between blocks

• draw_colored_turtle_square
  – Setup: set colors and begin to fill
  – Repeated Steps:
    • put the pen down, move forward, turn left
  – Pick pen up and fill in color

• draw_4_black_and_red
  – Setup: initialize fill color and pen color
  – Repeated Steps: change fill color,
    draw_colored_turtle_square, turn right, move forward
for_loop_checkerboard.py  3

• draw_4_black_and_red_4_times
  – repeated steps:
    • draw_4_black_and_red
    • move forward

• make_checkerboard
  – Setup: set turtle speed
  – Repeated Steps:
    • draw_4_black_and_red_4_times
    • Turn 180 degrees, move forward, turn 270 degrees, move forward turn 270 degrees
Woops

• While writing the checkerboard program, several mistakes led me to realize some interesting artistic possibilities
  – make_mistake1 – is the result of drawing bars with different angles

• This led me to experiment with different colors, more bars, etc., to make interesting patterns (wacky_bars, wacky_bars2)
About wacky_bars and wacky_bars2

- `random.random()` – chooses number between 0 and 1
- Colors in the turtle library consist of combinations of red, green and blue
  - Red has a value from 0 to 255
  - Green has a value from 0 to 255
  - Blue has a value from 0 to 255
  - 0,0,0 is black and 255,255,255 is white
- `math.ceil(random.random() * 255)` – choose a number between 1 and 255
- 3 such random numbers identifies a random color
More about wacky_bars

• Let red, green, blue be 3 numbers between 1 and 255

• Applying the following should produce contrasting colors:
  – red == (red - 128) % 255
  – green = (green - 128) % 255
  – blue = (blue - 128) % 255

• I am unaware of any scientific/artistic significance to this relation, e.g., these pairs of colors are not complementary