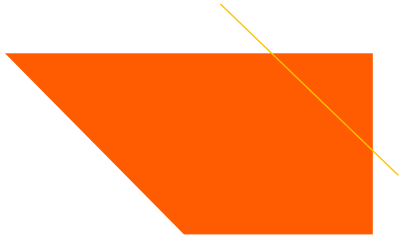
Introduction to Problem Solving in Python

COSI 10A



Class objectives

- Nested For Loops
- Managing Complexity

Review: `for` loop syntax

Syntax:

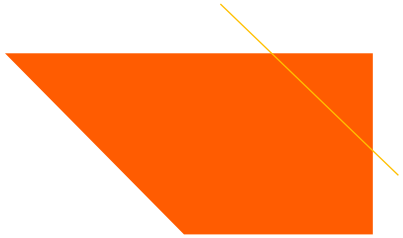
```
for variable in range (start, stop):
    statement
    statement
    ...
    statement
```

header

body

- Set the **loop variable** equal to the **start** value
- Repeat the following:
  - Check if the **variable** is **less than** the **stop**. If not, stop
  - Execute the **statement**s
  - Increase the variable's value by 1

```
for i in range(1, 6):     # repeat 5 times
    bake_cookies()
```



Review: Ways to create ranges

Range From	Description	Example	Numbers in Range
`range(max)`	Range from 0 (inclusive) to max (exclusive)	`range(5)`	0, 1, 2, 3, 4
`range(min, max)`	Range from min (inclusive) to max (exclusive)	`range(3, 7)`	3, 4, 5, 6
`range(min, max, step)`	Range from min (inclusive) to max (exclusive), increasing by step each time	`range(4, 22, 3)`	4, 7, 10, 13, 16, 19

More loops

- Suppose you want to print a multiplication table below:

1	2	3	4
2	4	6	8
3	6	9	12
4	8	12	16
5	10	15	20

- Do we like that?

```
def main():
    for x in range (1, 5):
        print(1 * x, end="\t")
    print()

    for x in range (1, 5):
        print(2 * x, end="\t")
    print()

    for x in range (1, 5):
        print(3 * x, end="\t")
    print()

    for x in range (1, 5):
        print(4 * x, end="\t")
    print()

    for x in range (1, 5):
        print(5 * x, end="\t")
    print()
main()
```

Nested `for` loop syntax

Syntax:

```
for variable1 in range (start, stop):
      for variable2 in range (start, stop):
            statement
            statement
            ...
            statement
      statement
      …
      statement
```

Nested loops

1	2	3	4
2	4	6	8
3	6	9	12
4	8	12	16
5	10	15	20

```
def main():
    for i in range(1, 6):
        for j in range(1, 5):
            print(i * j, end="\t")
        print()
main()
```

Exercise 1 (v.0)

- Write a for loop that produces the following output:

```
#
##
###
####
#####
```

Exercise 1 (v.0)

- Write a for loop that produces the following output:

```
#
##
###
####
#####
```

```
# print triangular figure w/string multiplication
for i in range(1, 6):
        print("#" * i)
```

Exercise 1 (v.1)

- Write a for loop that produces the following output:

```
#
##
###
####
#####
```

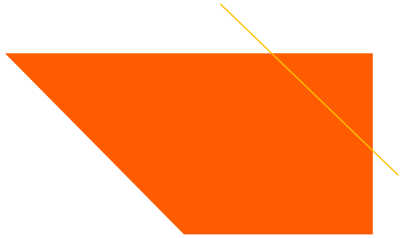
```
# print triangular figure w/out string multiplication
for i in range(1, 6):
    for j in range(i):
        print("#", end="")
    print()
```

Exercise 1 (v.2)

- Write a for loop that produces the following output:

```
#
##
###
####
#####
```

```
# print triangular figure w/out string multiplication
for i in range(1, 6):
    for j in range(1, i+1):
        print("#", end="")
    print()
```



Managing complexity

- When our programs solve more complex tasks
- Use of multiple functions
- "*Controlling complexity is the essence of computer programming*" - Brian Kernighan

- Scope : a tool to control interreference of different parts of our programs
  - Local
  - Global

Drawing complex figures (ASCII art) – Managing complexity

- Use nested `for` loops to produce the following output

```
#================#
|       <><>     |
|     <>....<>   |
|   <>........<> |
|<>............<>|
|<>............<>|
|   <>........<> |
|     <>....<>   |
|       <><>     |
#================#
```

Development strategy:

- Recommendations for managing complexity:
  1. Design the program (think about steps or methods needed).
     - write an English description of steps required (pseudo-code)
     - use this description to decide the functions
  2. Create a table of patterns of characters
     - use table to write your `for` loops

Pseudo-Code

- **pseudo-code**: An English description of an algorithm.

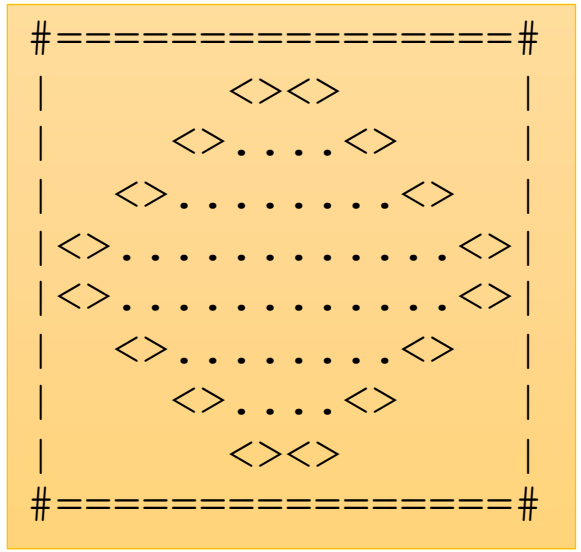
- Example: Drawing a 12 wide by 7 tall box of stars

```
print 12 stars.
for (each of 5 lines) :
    print a star.
    print 10 spaces.
    print a star.
print 12 stars.
```

```
************
*          *
*          *
*          *
*          *
*          *
************
```

Drawing complex figures (ASCII art)

- Write an English description of steps required (**pseudocode**)



1. Line

   # , 16 =, #

2. Top half

   |
   spaces (decreasing)
   <>
   dots (increasing)
   <>
   spaces (same as above)
   |

3. Bottom half (top half upside-down)

4. Line

   # , 16 =, #

ASCII art (v. 1)

```
def main():
    line()
    top_half()
    bottom_half()
    line()

def line():

def top_half():

def bottom_half():

main()
```

1. Line

   # , 16 =, #

2. Top half

   |
   spaces (decreasing)
   <>
   dots (increasing)
   <>
   spaces (same as above)
   |

3. Bottom half (top half upside-down)

4. Line

   # , 16 =, #

ASCII art (v. 2)

```
def main():
    line()
    top_half()
    bottom_half()
    line()

def line():
    print("#", end='')
    for i in range(16):
        print("=", end='')
    print("#")

def top_half():

def bottom_half():

main()
```

1. Line

 # , 16 =, #

2. Top half

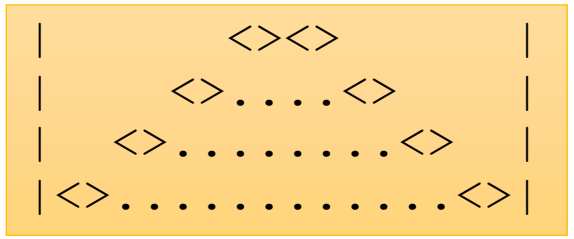
|
spaces (decreasing)
<>
dots (increasing)
<>
spaces (same as above)
|

3. Bottom half (top half upside-down)

4. Line

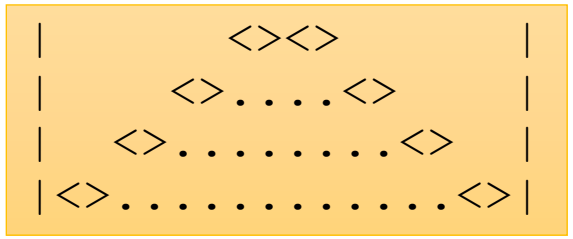
, 16 =,

ASCII art: Top half



line	spaces
1	6
2	4
3	2
4	0

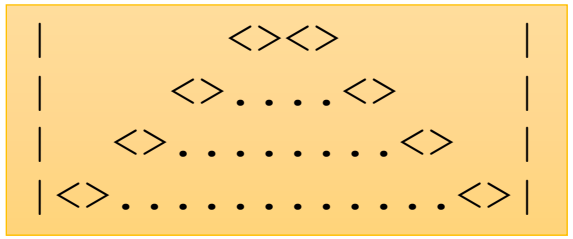
ASCII art: Top half



line	spaces	(-2 * line) + 8
1	6	6
2	4	4
3	2	2
4	0	0

```
def top_half():
    for line in range(1, 5):
        print("|", end="")
        for space in range(line * -2 + 8):
            print(" ", end="")
```

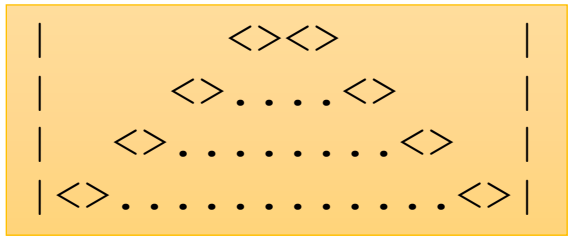
ASCII art: Top half



line	spaces	(-2 * line) + 8	dots
1	6	6	0
2	4	4	4
3	2	2	8
4	0	0	12

```
def top_half():
    for line in range(1, 5):
        print("|", end="")
        for space in range(line * -2 + 8):
            print(" ", end="")
```

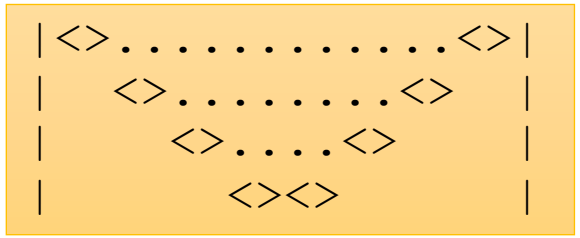
ASCII art: Top half



line	spaces	(-2 * line) + 8	dots	4 * line - 4
1	6	6	0	0
2	4	4	4	4
3	2	2	8	8
4	0	0	12	12

```
def top_half():
    for line in range(1, 5):
        print("|", end="")
        for space in range(line * -2 + 8):
            print(" ", end="")
        print("<>", end="")
        for dot in range(line * 4 - 4):
            print(".", end="")
        print("<>", end="")
        for space in range(line * -2 + 8):
            print(" ", end="")
        print("|")
```

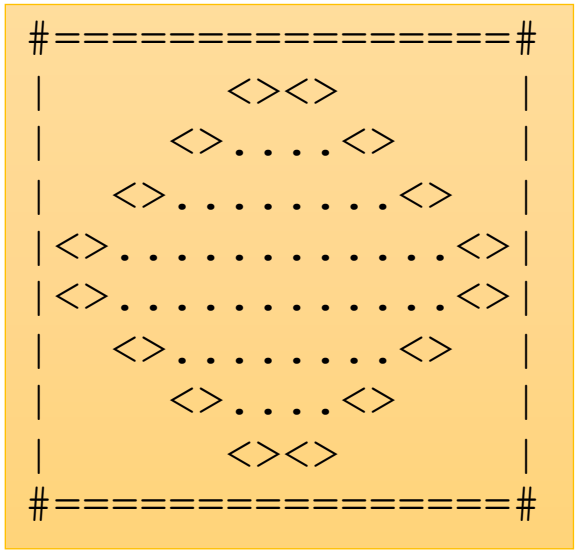
ASCII art: Bottom half



line	spaces	(-2 * line) + 8	dots	4 * line - 4
1	6	6	0	0
2	4	4	4	4
3	2	2	8	8
4	0	0	12	12

```
def bottom_half():
    for line in range(4, 0, -1):
        print("|", end="")
        for space in range(line * -2 + 8):
            print(" ", end="")
        print("<>", end="")
        for dot in range(line * 4 - 4):
            print(".", end="")
        print("<>", end="")
        for space in range(line * -2 + 8):
            print(" ", end="")
        print("|")
```

ASCII art




```
def main():
    line()
    top_half()
    bottom_half()
    line()
def line():
    print("#", end='')
    for i in range(16):
        print("=", end='')
    print("#")
def top_half():
    for line in range(1, 5):
        print("|", end="")
        for space in range(line * -2 + 8):
            print(" ", end="")
        print("<>", end="")
        for dot in range(line * 4 - 4):
            print(".", end="")
        print("<>", end="")
        for space in range(line * -2 + 8):
            print(" ", end="")
        print("|")
def bottom_half():
    for line in range(4, 0, -1):
        print("|", end="")
        for space in range(line * -2 + 8):
            print(" ", end="")
        print("<>", end="")
        for dot in range(line * 4 - 4):
            print(".", end="")
        print("<>", end="")
        for space in range(line * -2 + 8):
            print(" ", end="")
        print("|")
main()
```