Introduction to Problem Solving in Python

COSI 10A

Class objectives

- Managing Complexity
  - Constants
- Parameters

Review: Nested `for` loop

Syntax:

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

Review: 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

Review: 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 =, #

Review: 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="")
```

Review
: 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()
```

Scaling the ASCII art

- Let's modify our Mirror program so that it can scale

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

size = 4

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

size = 3

- We'd like to structure the code so we can scale the figure by changing the code in just one place

Constants

- A **constant** is a fixed value visible to the whole program (global scope)
  - The value should only be set at declaration; shouldn't be reassigned

Syntax: **name = value**

name is usually in ALL_UPPER_CASE

Examples:

```
DAYS_IN_WEEK = 7
INTEREST_RATE = 3.5
SSN = 658234569
```

Constants

- What equation would cause the code to print: `2 7 12 17 22`
- To see patterns, make a table of `SIZE` and the numbers
  - Each time `SIZE` goes up by 1, the number should go up by 5
  - But `SIZE * 5` is too great by 3, so we subtract 3

`SIZE`	number to print	`5 * SIZE`	`5 * SIZE - 3`
1	2	5	2
2	7	10	7
3	12	15	12
4	17	20	17
5	22	25	22

Scaling the ASCII art

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

size = 4

```
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("|")
```

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

size = 3

SIZE	line	spaces		dots	
4	1,2,3,4	6,4,2,0	line * -2 + **8**	0,4,8,12	line * 4 - 4
3	1,2,3	4,2,0	line * -2 + **6**	0,4,8	line * 4 – 4

ASCII art: Top half/w constant

```
|        <><>        |
|     <>....<>       |
|   <>........<>     |
|<>............<>|
```

line	spaces	dots
1	6	0
2	4	4
3	2	8
4	0	12

```python
SIZE = 4;

# Prints the expanding pattern of <> for the top half of the figure
def top_half():
    for line in range(1, SIZE+1):
        print("|", end="")
        for space in range(line * -2 + (2*SIZE)):
            print(" ", end="")
        print("<>", end="")
        for dot in range(line * 4 - 4):
            print(".", end="")
        print("<>", end="")
        for space in range(line * -2 + (2*SIZE)):
            print(" ", end="")
        print("|")

top_half()
```

ASCII art: Top half/w constant

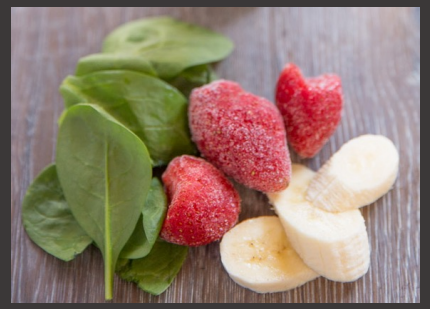
```
|     <><>     |
|   <>....<>   |
|<>........<>|
```

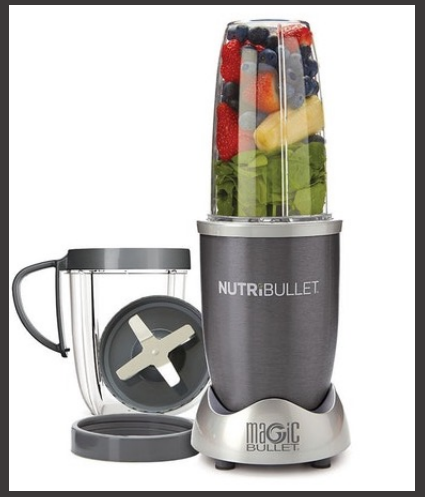
line	spaces	dots
1	4	0
2	2	4
3	0	8

```python
SIZE = 3;

# Prints the expanding pattern of <> for the top half of the figure
def top_half():
    for line in range(1, SIZE+1):
        print("|", end="")
        for space in range(line * -2 + (2*SIZE)):
            print(" ", end="")
        print("<>", end="")
        for dot in range(line * 4 - 4):
            print(".", end="")
        print("<>", end="")
        for space in range(line * -2 + (2*SIZE)):
            print(" ", end="")
        print("|")

top_half()
```

Parameters

Parameters





Parameters

- Suppose you want to write a program that draws box-shaped figures, e.g. a box of size 6x6

```
******
*....*
*....*
*....*
*....*
******
```

```
def main():
    draw_box()

def draw_box():
        print("*" * 6)
        for line in range(4):
                print("*", "." * 4, "*", sep='')
        print("*" * 6)

main()
```

Parameters

- Now suppose we want to draw three boxes of various sizes (6x6, 9x9, 4x4)

```
******
*....*
*....*
*....*
*....*
******
*********
*.......*
*.......*
*.......*
*.......*
*.......*
*.......*
*.......*
*********
****
*..*
*..*
****
```

Do we like this? ->

```
def main():
    draw_box1()
    draw_box2()
    draw_box3()

def draw_box1():
        print("*" * 6)
        for line in range(4):
                print("*", "." * 4, "*", sep='')
        print("*" * 6)

def draw_box2():
    print("*" * 9)
    for line in range(7):
        print("*", "." * 7, "*", sep='')
    print("*" * 9)

def draw_box3():
    print("*" * 4)
    for line in range(2):
        print("*", "." * 2, "*", sep='')
    print("*" * 4)

main()
```

Parameters

- Let's use parameters!!!
- A **parameter** is a value passed to a function by its caller

Function

Caller

Function

Caller

```
def main():
    draw_box()

def draw_box():
        print("*" * 6)
        for line in range(4):
            print("*", "." * 4, "*", sep='')
        print("*" * 6)

main()
```

Parameters

- A **parameter** is a value passed to a function by its caller

- When **declaring** the function, we will state that it requires a parameter indicating the size of the box
- When **calling** the function, we will specify the size

Declaring a parameter

- Declaring a parameter means stating that a function requires a parameter in order to run

Syntax:

```
def <name> (<name>):
    <statement>(s)
```

Example:

```
def say_password(code):
    print("The password is:", code)
```

- When `say_password` is called, the caller <u>must</u> specify the **code** to print

Passing a parameter

- Passing a parameter means calling a function specifying the value for its parameters

Syntax: ***<name>*** (***<expression>***)

Example:

```
say_password(42)
say_password(12345)
```

Output:

```
The password is 42
The password is 12345
```

Passing a parameter

```
def main():
    say_password(3342)
    print()
    say_password("abracadabra")
    print()
    say_password(89.6)


def say_password(code):
    print("The password is:", code)

main()
```

Parameters

- Now suppose we want to draw three boxes of various sizes (6x6, 9x9, 4x4)

```
******
*....*
*....*
*....*
*....*
******
*********
*.......*
*.......*
*.......*
*.......*
*.......*
*.......*
*.......*
*********
****
*..*
*..*
****
```

Redundant

```
def main():
    draw_box1()
    draw_box2()
    draw_box3()

def draw_box1():
        print("*" * 6)
        for line in range(4):
                print("*", "." * 4, "*", sep='')
        print("*" * 6)

def draw_box2():
    print("*" * 9)
    for line in range(7):
        print("*", "." * 7, "*", sep='')
    print("*" * 9)

def draw_box3():
    print("*" * 4)
    for line in range(2):
        print("*", "." * 2, "*", sep='')
    print("*" * 4)

main()
```

Parameters

- Now suppose we want to draw three boxes of various sizes (6x6, 9x9, 4x4)

```
******
*....*
*....*
*....*
*....*
******
```

```
def main():
    draw_box(6)
    draw_box(9)
    draw_box(4)

def draw_box(size):
        print("*" * size)
        for line in range(size-2):
                print("*", "." * (size-2), "*", sep='')
        print("*" * size)


main()
```

6

6 size

Parameters

- Now suppose we want to draw three boxes of various sizes (6x6, 9x9, 4x4)

```
******
*....*
*....*
*....*
*....*
******
*********
*.......*
*.......*
*.......*
*.......*
*.......*
*.......*
*.......*
*********
```

```
def main():
    draw_box(6)
    draw_box(9)
    draw_box(4)

def draw_box(size):
        print("*" * size)
        for line in range(size-2):
                print("*", "." * (size-2), "*", sep='')
        print("*" * size)


main()
```

9

9 size

Parameters

- Now suppose we want to draw three boxes of various sizes (6x6, 9x9, 4x4)

```
******
*....*
*....*
*....*
*....*
******
*********
*.......*
*.......*
*.......*
*.......*
*.......*
*.......*
*.......*
*********
****
*..*
*..*
****
```

```
def main():
    draw_box(6)
    draw_box(9)
    draw_box(4)

def draw_box(size):
        print("*" * size)
        for line in range(size-2):
                print("*", "." * (size-2), "*", sep='')
        print("*" * size)


main()
```

4

4 size

- A parameter sends a value into a function