CSC 111:
Intro to Computer Science through Programming

Spring 2017
Prof. Sara Mathieson

Admin

- Homework 5 due Tues after spring break
- Lab next week will be like TA hours (optional time to ask questions and work on homework/studying)
- Fill out poll for lab next week on Piazza!
- Practice midterms will be returned soon (this weekend or Monday)
- Note: quizzes are a very small (about 10%) fraction of your overall homework grade (I know it's easy to miss a few questions and get a low score)

Outline: 3/3

- Recap / finish: reading and writing files
- Go over Homework 2
- Go over midterm practice questions and Quiz 4
- Start: Midterm review

Recap reading/writing files

Informal quiz (discuss with a partner)

1) What is **int(line)** doing?

2) What does **append** return?

3) What are the similarities and differences between **append** and the commented way of adding to a list?

```
import list_range

def main():

    num_lst = []

    number_file = open("numbers.txt","r")
    for line in number_file:

        line = line.strip()
        num = int(line)

        #num_lst = num_lst + [num]
        num_lst.append(num)
    number_file.close()

    print(num_lst)

    my_min = list_range.minimum(num_lst)
    print(my_min)
    my_max = list_range.maximum(num_lst)
    print(my_max)

main()
```

Informal quiz (discuss with a partner)

1) What is **int(line)** doing?

Converting ("casting") a string to an int

2) What does **append** return?

3) What are the similarities and differences between **append** and the commented way of adding to a list?

```
import list_range

def main():

    num_lst = []

    number_file = open("numbers.txt","r")
    for line in number_file:

        line = line.strip()
        num = int(line)

        #num_lst = num_lst + [num]
        num_lst.append(num)
    number_file.close()

    print(num_lst)

    my_min = list_range.minimum(num_lst)
    print(my_min)
    my_max = list_range.maximum(num_lst)
    print(my_max)

main()
```

Informal quiz (discuss with a partner)

1) What is **int(line)** doing?

Converting ("casting") a string to an int

2) What does **append** return?

Nothing!

3) What are the similarities and differences between **append** and the commented way of adding to a list?

```
import list_range

def main():

    num_lst = []

    number_file = open("numbers.txt","r")
    for line in number_file:

        line = line.strip()
        num = int(line)

        #num_lst = num_lst + [num]
        num_lst.append(num)
    number_file.close()

    print(num_lst)

    my_min = list_range.minimum(num_lst)
    print(my_min)
    my_max = list_range.maximum(num_lst)
    print(my_max)

main()
```

Informal quiz (discuss with a partner)

1) What is **int(line)** doing?

Converting ("casting") a string to an int

2) What does **append** return?

Nothing!

3) What are the similarities and differences between **append** and the commented way of adding to a list?

```
import list_range

def main():

    num_lst = []

    number_file = open("numbers.txt","r")
    for line in number_file:

        line = line.strip()
        num = int(line)

        #num_lst = num_lst + [num]
        num_lst.append(num)
    number_file.close()

    print(num_lst)
```

The first way concatenates (+) two lists to create a new list with one more element. The original list variable is then reassigned to the value of this new list. Append mutates (modifies) the original list to have one more element at the end, but does not return anything.

Three more useful methods…

- **.strip()** for string: removes whitespace at the beginning and end of a string
- **.append(element)** for list: add an element to a list, replaces our idea of **lst = lst + [element]**
- **.index(element)** for a list or string:

  returns the index of the element

```
>>> lst = [17,10,1]
>>> lst.index(10)
1
>>> lst.index(1)
2
>>>
>>> string = "Spring break is coming soon"
>>> string.index("p")
1
>>> string.index("i")
3
>>> string.index("break")
7
>>> string.index("i",4)
13
>>> string.index("i",13)
13
>>> string.index("i",14)
19
>>>
```

Homework 2 Examples

(selected by Aditya)

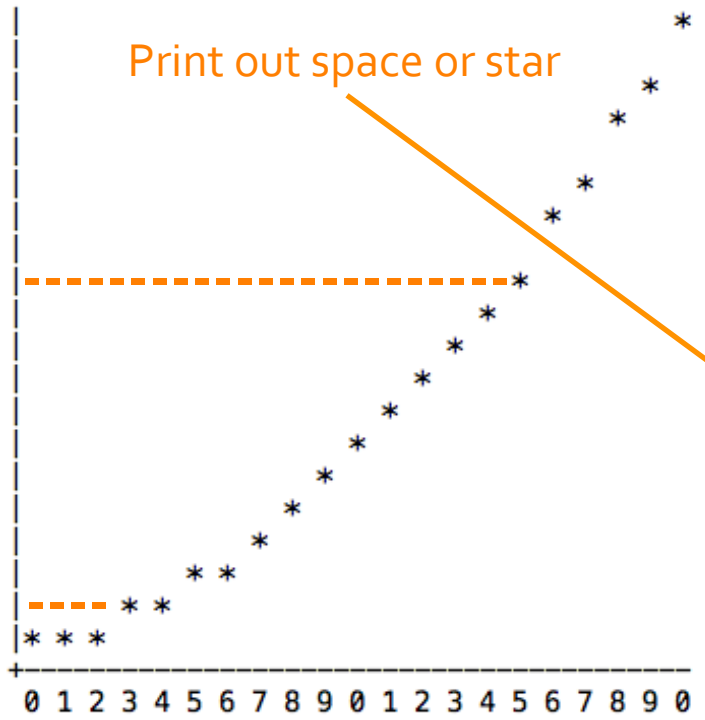
Homework 2 Example: Gabrielle Sholars

```
# For-loop that prints a table based on data given by user.
table_level = float(0)
for k in range (yrs+1):
    table_yrs = str(k)
    table_rate = round(rate + accel * k, 2)

    print("year=" + table_yrs, "rate=" + str(table_rate),
          "level=" + str(table_level))

    # I place the level calculation after the print statement
    # in order to generate the inital level of 0.0 @ year 0.
    table_level = round(table_level + table_rate, 2)

print( )

#For-loop that prints graph based on data displayed in table.
graph_level = float(0)
for i in range(yrs+1):
    graph_accel = int(round(graph_level/5))
    if i == 10:          #To generate solely the unit digit (symmetry)
        i = i % 10
    if i > 10:           #To generate solely the unit digit (symmetry)
        i = i % 10
        graph_accel = graph_accel + 1
    print(str(i) + "|" + " " * graph_accel + "*")
    graph_level = round((graph_level + rate + accel * i), 1)

#border of graph
print(" +" + "-" * (int(graph_accel) + 1))
print( )
```

Homework 2 Example: Elizabeth Muirhead

```
level = 0
stored_lev = []

# Construct chart
for i in range (years+1):

    # Calculate rate
    chart_rate = ((rate+(i*acceleration)))

    # Store values of level in a list for graph
    stored_lev.append(level)

    # Print the chart, round values
    print("years=", i, " ", "rate=", round(chart_rate, 1)," ",
          "level=", round(level, 1), sep = "")

    # Keep track of level values
    level = level+chart_rate

# Construct Graph
print ("")

for i in range (years+1):

    # Take levels that have been stored in my list and
    # convert them vaules that will be used to space the graph
    chart_num = stored_lev[i]

    chart_num = int(round(stored_lev[i]/5))

    # Print graph
    print( i%10,"|", " "*chart_num, "*", sep = "")

# Print bottom of graph
print(" +", "-"*(chart_num+1), sep = "")
```

Homework 2 Example: Eve Xu




```
# reset the original level and space_list
level = 0.0
space_list = []

# second for-loop for graph
for j in range(year + 1):
    rate = round(curr_rate + acc_rate * j, 1)
    # each space represents 5 mm
    space = round(level / 5)
    space_list = space_list + [space]
    print(str(j % 10) + "|" + " " * space + "*")
    level = round(level + rate, 1)

# the other axis
print(" " + "+" + "-" * (space + 1) )

# space between two graphs
print()

# thirld for loop for rotated graph
# a for-loop to set the vertical axis
for m in range(space,-1,-1):
    print("\n"+"|", end = "")

    # a for-loop for points
    for k in range(year+1):
        if(space_list[k] == m):
            print("*", end = " ")

        else:
            print(" ", end = " ")


# the base axis
print("\n"+"+" + "-" * (2 * year + 1))
```

Homework 2 Example: Garcia Sun


```
# from here is the challenge
# preserve the last value of the sea level,
# aka the max sea level to create the y-axis;
# print a line to split the horizontal graph
# and the vertical one
y = round(level_list[-1]/5)
print()

# use a for loop to create a horizontal graph: using the booleans
# to see if we should print space or print * and save them into
# the list called row for later print
for i in range(y+1):
    print()
    row_list = ["|"]
    for j in range(years+1):
        if round(level_list[j]/5) == y-i:
            row_list = row_list + ["* "]
        else:
            row_list = row_list + ["  "]
    for k in row_list:
        print(k, end = "")
print("\n"+"+"+"-"*(2*years+1))
for i in range(years+1):
    print(" ", end = "")
    print(str(i%10),end = "")
```

Midterm review

Practice question 1

- What does this code produce?

```
def print_min(num1,num2):
    if num1 < num2:
        print("The first number is the min")
    if num2 < num1:
        print("The second number is the min")
    else:
        print("They are equal")


def main():
    num1 = 5
    num2 = 7
    print_min(num2, num1)

main()
```

Practice question 1

- What does this code produce?

```
def print_min(num1,num2):
    if num1 < num2:
        print("The first number is the min")
    if num2 < num1:
        print("The second number is the min")
    else:
        print("They are equal")


def main():
    num1 = 5
    num2 = 7
    print_min(num2, num1)

main()
```

```
>>>
The second number is the min
>>>
```

Practice question 2

- What does this code produce?

```
def print_min(num1,num2):
    if num1 < num2:
        print("The first number is the min")
    if num2 < num1:
        print("The second number is the min")
    else:
        print("They are equal")


def main():
    num1 = 5
    num2 = 1
    print_min(num2, num1)

main()
```

Practice question 2

- What does this code produce?

```
def print_min(num1,num2):
    if num1 < num2:
        print("The first number is the min")
    if num2 < num1:
        print("The second number is the min")
    else:
        print("They are equal")


def main():
    num1 = 5
    num2 = 1
    print_min(num2, num1)

main()
```

```
>>>
The first number is the min
They are equal
>>>
```

Practice question 2

- What does this code produce?

elif (!!)

```
def print_min(num1,num2):
    if num1 < num2:
        print("The first number is the min")
    if num2 < num1:
        print("The second number is the min")
    else:
        print("They are equal")


def main():
    num1 = 5
    num2 = 1
    print_min(num2, num1)

main()
```

```
>>>
The first number is the min
They are equal
>>>
```

Practice Midterm Part A

```
def main():

    # ask the user for a name and a number
    name = input("Enter your name: ")
    num  = eval(input("Enter a number: "))

    # print out the name, number of times increasing
    for i in range(1,num+1):
        print(name*i)

    # print out the name, number of times decreasing
    for i in range(num-1,0,-1):
        print(name*i)

main()
```

Practice Midterm Part B

- Main issues:
  - Boolean variables
  - Variable scope

DEMO

```
def main():
    string = input("Enter a string: ")

    if palindrome(string):
        print("\n>> " + string, "is a palindrome\n")
        for i in range(len(string)):

            if i == 0 or i == len(string)-1:
                print(string)
            else:
                stars = "*" * (len(string)-2)
                print(string[i] + stars + string[i])

    else:
        print("\n>> " + string, "is not a palindrome\n")
        print(string + reverse(string))

main()
```

Quiz 4, Question 2

Question 2

Correct

1.00 points out of 1.00

```
num1 = 3
num2 = 3
def printMax( num1, num2 ):
        if num1 >  num2:
                print( str(num1), 'is the max' )
        elif num2 > num1:
                print( str(num2), 'is the max' )
        else:
                print( 'they are equal')
        return max
```

What happens when I **call** `printMax()` with the **arguments 3** and **4**?

Quiz 4, Question 2

Question 2

Correct

1.00 points out of 1.00

```
num1 = 3
num2 = 3
def printMax( num1, num2 ):
        if num1 >  num2:
                print( str(num1), 'is the max' )
        elif num2 > num1:
                print( str(num2), 'is the max' )
        else:
                print( 'they are equal')
        return max
```

What happens when I **call** `printMax()` with the **arguments 3** and **4**?

Select one:

- 3 is the max
- they are equal
- 4 is the max ✓
- None of the above

Quiz 4, Question 3

Question 3

Correct

1.00 points out of 1.00

```
def greeting( person ):
        greeting = 'Hi ' + person


greeting('Chloe')
greeting('Scarlet')
greeting('Mathew')
```

What is printed when you execute this piece of code?

Quiz 4, Question 3

Question 3

Correct

1.00 points out of 1.00

```
def greeting( person ):
        greeting = 'Hi ' + person


greeting('Chloe')
greeting('Scarlet')
greeting('Mathew')
```

What is printed when you execute this piece of code?

Select one:

- a. Chole
  Scarlet
  Mathew
- b. None of the above ✓
- c. Mathew
- d. Error
- e. Mathew
  Scarlet
  Chole

Exercise: boolean practice

- TODO: write a function to determine whether or not a string has a repeated letter

```
>>> repeated_letter("hello")
True
>>> repeated_letter("goodbye")
True
>>> repeated_letter("spring")
False
>>>
>>> repeated_letter("boolean")
True
>>> repeated_letter("fall")
True
>>> repeated_letter("break")
False
>>>
```