CSC 111:
Intro to Computer Science through Programming

Spring 2017
Prof. Sara Mathieson

Admin

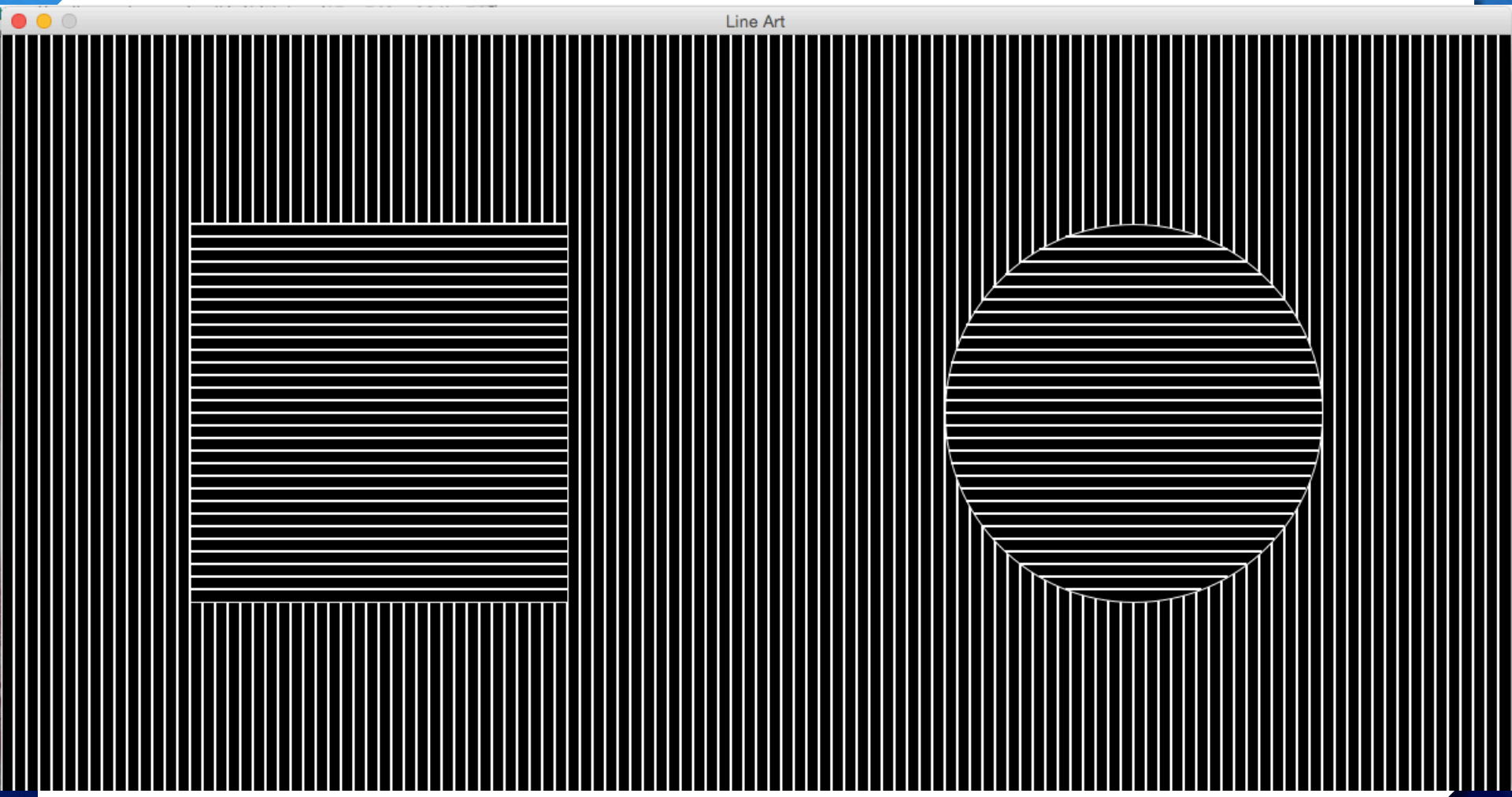
- Homework 6 is due March 28
- No transcripts for graphics (screenshots instead)
- I will be away next week at a conference, but there will still be class, lab, and homework as usual (but no office hours)

Outline: 3/24

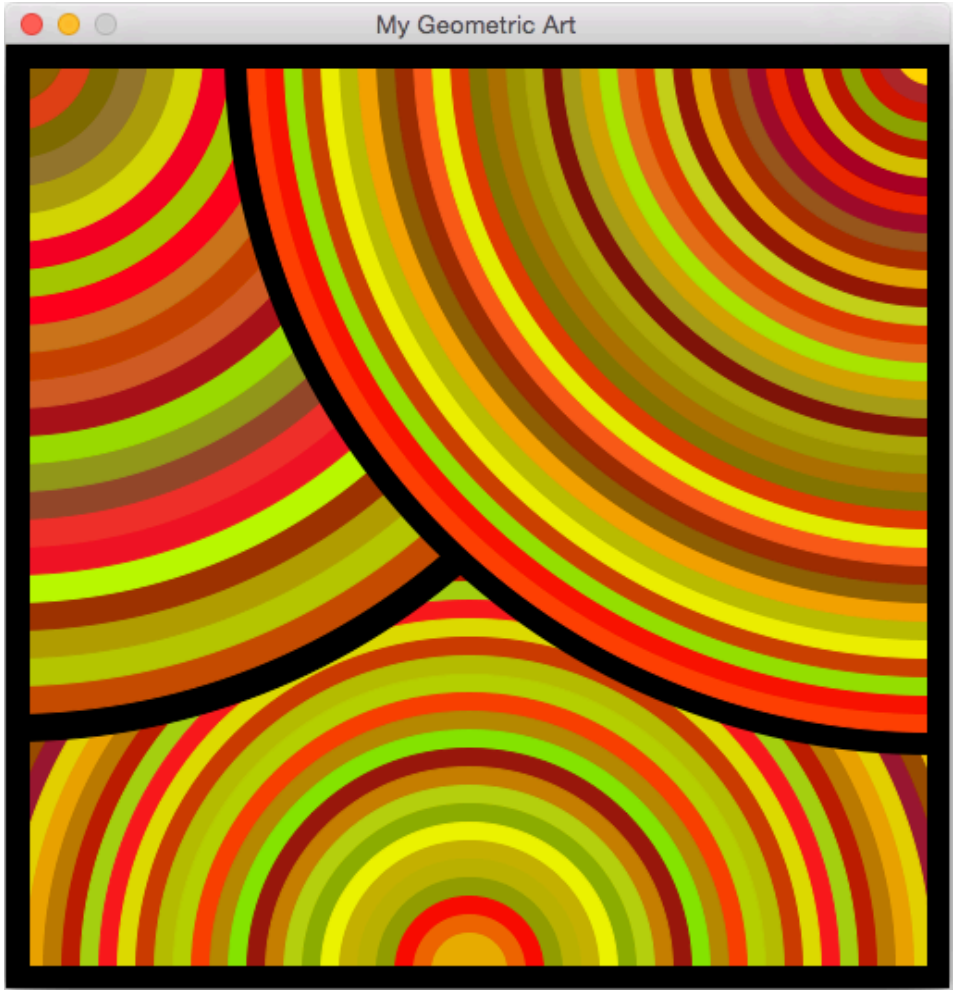
- Recap last time
- Midterm feedback
- Continue: animated graphics
- User input in graphics

Recap
(+ snow examples)

Chelsey



Tani



Guess my number with while loop

```
import random

def main():

    # ask the user for a random number
    number = eval(input("Enter a number: "))

    # start off with an initial guess
    guess = random.randint(1,100)
    print(guess)

    # while the guess is not equal to the user's number, guess again
    while guess != number:
        guess = random.randint(1,100)
        print(guess)

    # after the while loop is over, the computer must
    # have guessed the number!
    print("You guessed it!")

main()
```

Midterm Feedback

Part 1

(a) What numbers are printed when the following code is executed?

(b) What numbers are printed when the following code is executed?

Part 1

(a) What numbers are printed when the following code is executed?

```
>>> for i in range(6,-8,-2):
        print(i)


6
4
2
0
-2
-4
-6
```

(b) What numbers are printed when the following code is executed?

```
>>> for i in range(-3,-1):
        print(i)


-3
-2
```

Part 1

(c) Describe the issue with the following code, which is attempting to return the sum of the numbers in a list.

```
def my_sum(lst):
    s = 0
    for i in lst:
        s = s + lst[i]
    return s
```

Part 1

(c) Describe the issue with the following code, which is attempting to return the sum of the numbers in a list.

```
def my_sum(lst):
    s = 0
    for i in lst:
        s = s + lst[i]
    return s
```

i is an element of the list, not an index, so trying to index in this way will produce errors (changing i to elem or something similar would help).

```
    s = s + lst[i]
IndexError: list index out of range
```

```
    s = s + lst[i]
TypeError: list indices must be integers, not float
```

Part 1

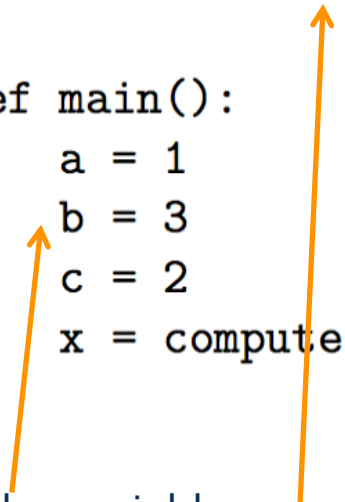
(d) If I invoke the main method below, what will be the value of **x**?

```
def compute(a,b,c):
    return a*(b-c)

def main():
    a = 1
    b = 3
    c = 2
    x = compute(b,c,a)
```

Part 1

(d) If I invoke the main method below, what will be the value of x?

```
def compute(a,b,c):
    return a*(b-c)

def main():
    a = 1
    b = 3
    c = 2
    x = compute(b,c,a)
```



These a,b,c variables are completely separate from these a,b,c variables.

```
>>> compute(3,2,1)
>>> 3*(2-1)
x = 3
```

Part 1

(3) (e) What is wrong with this attempt to read a file? How could you modify the variable names to help prevent this type of error?

```
words = open("my_file.txt","r")
for text in words:
    x = words.count("a")
```

text

I would change:

- words → file
- text → line

- The main issue is that words is the file object, so we cannot call words.count("a"). Should be text.count("a").
- We should also close the file!

Part 1

(f) The following lines of code are executed sequentially in the shell. In the right column, write out what is printed *after* each line (some lines of code may not have any output). If a line produces an error, just write "error". Then circle all the lines that make use of *casting*.

Part 1

(f) The following lines of code are executed sequentially in the shell. In the right column, write out what is printed *after* each line (some lines of code may not have any output). If a line produces an error, just write "error". Then circle all the lines that make use of *casting*.

line of code	output
`>>> x = "3.14"`	/
`>>> float(x)`	3.14
`>>> x = x + 1`	error
`>>> y = float(x)`	/
`>>> y`	3.14
`>>> z = int(y)`	/
`>>> z`	3
`>>> type(z)`	<class 'int'>
`>>> type(y)`	<class 'float'>
`>>> type(x)`	<class 'str'>

Part 1

(g) Write a function in the box below that will take one parameter, a list of full names, and *modify* the list so that it only contains the first names. A few examples are shown below:

```
>>> lst1 = ["Katherine Johnson","Dorothy Vaughan","Mary Jackson"]
>>> first_name(lst1)
>>> lst1
['Katherine', 'Dorothy', 'Mary']
>>> lst2 = ["Logan Swanson","Artemis Metaxa","Val McCulloch","Zoe Kendall"]
>>> first_name(lst2)
>>> lst2
['Logan', 'Artemis', 'Val', 'Zoe']
```

Part 1

(g) Write a function in the box below that will take one parameter, a list of full names, and *modify* the list so that it only contains the first names. A few examples are shown below:

```
>>> lst1 = ["Katherine Johnson","Dorothy Vaughan","Mary Jackson"]
>>> first_name(lst1)
>>> lst1
['Katherine', 'Dorothy', 'Mary']
>>> lst2 = ["Logan Swanson","Artemis Metaxa","Val McCulloch","Zoe Kendall"]
>>> first_name(lst2)
>>> lst2
['Logan', 'Artemis', 'Val', 'Zoe']
```

```
def first_name(lst):
    for i in range(len(lst)):
        first_last = lst[i].split()
        lst[i] = first_last[0]
```

Part 2

The goal for this question is to write a function `capital_index(string)`, which returns the *index* of the first capital letter in a string. If there are no capital letters in the string, the function should print `"No capital letters found"`. Here are a few examples of this function in the shell:

```
>>> capital_index("Spring")
0
>>> capital_index("brEaK")
2
>>> capital_index("hello")
No capital letters found
>>> capital_index("computeR")
7
>>> capital_index("sciEnCE")
3
```

Part 2

The goal for this question is to write a function `capital_index(string)`, which returns the *index* of the first capital letter in a string. If there are no capital letters in the string, the function should print `"No capital letters found"`. Here are a few examples of this function in the shell:

```
>>> capital_index("Spring")
0
>>> capital_index("brEaK")
2
>>> capital_index("hello")
No capital letters found
>>> capital_index("computeR")
7
>>> capital_index("sciEnCE")
3
```

```
# returns whether or not a single letter is a capital
# example: "a" will return False and "A" will return True
def is_capital(letter):
    if letter == letter.upper():
        return True
    return False

def capital_index(string):
    for i in range(len(string)):
        if is_capital(string[i]):
            return i
    print("No capital letters found")
```

Part 3

```
import random

def mystery_method1(n):
    r_lst = []
    for i in range(n):
        r_lst.append(random.randint(0,19))
    return r_lst

def mystery_method2(lst):
    for j in range(len(lst)-1):
        if lst[j] > lst[j+1]:
            return False
    return True

def main():
    num = eval(input("Enter a number: "))
    my_lst = mystery_method1(num)
    result = mystery_method2(my_lst)

    print(my_lst)
    print(result)

main()
```

Part 3

```
import random

def mystery_method1(n):
    r_lst = []
    for i in range(n):
        r_lst.append(random.randint(0,19))
    return r_lst

def mystery_method2(lst):
    for j in range(len(lst)-1):
        if lst[j] > lst[j+1]:
            return False
    return True

def main():
    num = eval(input("Enter a number: "))
    my_lst = mystery_method1(num)
    result = mystery_method2(my_lst)

    print(my_lst)
    print(result)

main()
```

Formal parameters:
n,lst

Actual parameters:
num, my_lst

randint:
function

Goal: find out if the list is in ascending order (i.e. sorted low to high)

Part 4

The following examples show an inverted triangle being printed after the user enters an integer. Write and call a main method to produce this result. Notice that if the number is 3, there are 7 stars in the first row, and if the number is 5, there are 11 stars in the first row. Your main method should be general for any integer.

Part 4

The following examples show an inverted triangle being printed after the user enters an integer. Write and call a main method to produce this result. Notice that if the number is 3, there are 7 stars in the first row, and if the number is 5, there are 11 stars in the first row. Your main method should be general for any integer.

```
def main():
    n = eval(input("Enter a number: "))
    print("*"*(2*n+1))

    for i in range(1,n):
        print(" "*i + "*" + " "*(2*(n-i)-1) + "*")
    print(" "*n + "*")

main()
```

Part 5

In class we have seen how to swap the values of two variables, and in homework we have seen how to swap the values of two elements in a list. In this question, the goal is to swap the values of *three* variables, so that each variable ends up with the value of the "previous" variable. For example, if `x = 6`, `y = 3`, and `z = 1`, then the end result should be `x = 1`, `y = 6`, and `z = 3`.

(a) The following code shows a first attempt at this process. Fill in the table below, showing what will happen *after* each line is executed. The first row has been filled in with the initial values from the example above.

Part 5

In class we have seen how to swap the values of two variables, and in homework we have seen how to swap the values of two elements in a list. In this question, the goal is to swap the values of *three* variables, so that each variable ends up with the value of the "previous" variable. For example, if `x = 6`, `y = 3`, and `z = 1`, then the end result should be `x = 1`, `y = 6`, and `z = 3`.

(a) The following code shows a first attempt at this process. Fill in the table below, showing what will happen *after* each line is executed. The first row has been filled in with the initial values from the example above.

code	x	y	z	temp1	temp2
	6	3	1	-	-
`temp1 = x`	6	3	1	6	-
`temp2 = y`	6	3	1	6	3
`x = z`	1	3	1	6	3
`y = x`	1	1	1	6	3
`z = y`	1	1	1	6	3

Part 5

(d) Would it be possible to write a *function* to swap three variables? i.e. would it be possible to write a function that produces the output below? Explain your answer and reasoning, using the concept of mutable vs. immutable types.

```
>>> x = 6
>>> y = 3
>>> z = 1
>>> three_way_swap(x,y,z)
>>> x
1
>>> y
6
>>> z
3
```

Part 5

(e) Write a function that will return a new string with the i^{th} character and j^{th} character swapped. Example: `string_swap("spring",1,5)` should return `"sgrinp"`. You may assume that i is less than j. Why must this function return a *new* string?

```
def string_swap(string, i, j):
```

Notes about grades

- This exam was very challenging!
- Think about the midterm not as a number, but as an opportunity for improvement
- Trajectory is very important
- Midterm is only worth 15%
- Grade on Moodle is lower bound for final hw/lab grade

Midterm curve

- 95-100: A+
- 90-94: A
- 85-89: A-
- 80-84: B+
- 70-79: B
- 65-69: B-
- 60-64: C+
- 55-59: C
- 50-54: C-
- < 50: not passing

Average: 77

Continue: animated graphics