CSC 111: Intro to Computer Science through Programming

Spring 2017
Prof. Sara Mathieson
Admin

- Sit somewhere new! Sit by someone you haven’t met yet
- Homework 4 due Tuesday (tomorrow)
- Office hours today 3-5pm in Ford 355 or across the hall
Outline: 2/27

- Recap last time (min/max/range)
- Different types of errors and how to debug
- Go over Lab 3
- Begin: reading and writing files
Recap
```python
def minimum(lst):
    my_min = lst[0]
    for i in range(len(lst)):
        if lst[i] < my_min:
            my_min = lst[i]
    return my_min

def maximum(lst):
    my_max = lst[0]
    for i in range(len(lst)):
        if lst[i] > my_max:
            my_max = lst[i]
    return my_max

def lst_range(lst):
    my_min = minimum(lst)
    my_max = maximum(lst)
    r = my_max - my_min
    return r

def main():
    my_lst = [17, 10, 1, 12, 5, 18, 15, 16, 6, 14]
    r = lst_range(my_lst)
    print("Range is", r)

main()
```
Informal Quiz (discuss with a partner)

1) Are strings mutable or immutable? What about lists? What are the implications of this for functions?

2) When analyzing error messages, should you start from:
   A. the first line number
   B. the last line number
   C. somewhere in the middle

3) Why was there no else case in our min/max functions?

4) If a function returns, what do we usually do with the result?

5) If a function doesn’t return, what does it usually do instead?
Informal Quiz (discuss with a partner)

1) Are strings mutable or immutable? What about lists? What are the implications of this for functions?

Strings are immutable, Lists are mutable. A function can modify a list without returning anything.

2) When analyzing error messages, should you start from:
   A. the first line number
   B. the last line number
   C. somewhere in the middle

3) Why was there no else case in our min/max functions?

4) If a function returns, what do we usually do with the result?

5) If a function doesn’t return, what does it usually do instead?
Informal Quiz (discuss with a partner)

1) Are strings mutable or immutable? What about lists? What are the implications of this for functions?

Strings are immutable, Lists are mutable. A function can modify a list without returning anything.

2) When analyzing error messages, should you start from:
   A. the first line number
   B. the last line number
   C. somewhere in the middle

3) Why was there no else case in our min/max functions?

4) If a function returns, what do we usually do with the result?

5) If a function doesn’t return, what does it usually do instead?
Informal Quiz (discuss with a partner)

1) Are strings mutable or immutable? What about lists? What are the implications of this for functions?

Strings are immutable, Lists are mutable. A function can modify a list without returning anything.

2) When analyzing error messages, should you start from:
   A. the first line number
   B. the last line number
   C. somewhere in the middle

3) Why was there no `else` case in our min/max functions?

   If the current number wasn’t less than the current min, no need to do anything or update anything.

4) If a function `returns`, what do we usually do with the result?

5) If a function doesn’t `return`, what does it usually do instead?
Informal Quiz (discuss with a partner)

1) Are strings mutable or immutable? What about lists? What are the implications of this for functions?

Strings are immutable, Lists are mutable. A function can modify a list without returning anything.

2) When analyzing error messages, should you start from:
   A. the first line number
   B. the last line number
   C. somewhere in the middle

3) Why was there no else case in our min/max functions?

   If the current number wasn’t less than the current min, no need to do anything or update anything.

4) If a function returns, what do we usually do with the result?

   We usually assign the value of the result to a variable, i.e. new_variable = my_function(params)

5) If a function doesn’t return, what does it usually do instead?
Informal Quiz (discuss with a partner)

1) Are strings mutable or immutable? What about lists? What are the implications of this for functions?

Strings are immutable, Lists are mutable. A function can modify a list without returning anything.

2) When analyzing error messages, should you start from:
   A. the first line number
   B. the last line number
   C. somewhere in the middle

3) Why was there no else case in our min/max functions?

If the current number wasn’t less than the current min, no need to do anything or update anything.

4) If a function returns, what do we usually do with the result?

We usually assign the value of the result to a variable, i.e.  
new_variable = my_function(params)

5) If a function doesn’t return, what does it usually do instead?

Modify something! (right now a list)
def minimum(lst):
    my_min = lst[0]
    for i in range(len(lst)):
        if lst[i] < my_min:
            my_min = lst[i]
    return my_min

def maximum(lst):
    my_max = lst[0]
    for i in range(len(lst)):
        if lst[i] > my_max:
            my_max = lst[i]
    return my_max

def lst_range(lst):
    my_min = minimum(lst)
    my_max = maximum(lst)
    r = my_max - my_min
    return r

def main():
    my_lst = [17, 10, 1, 12, 5, 18, 15, 16, 6, 14]
    r = lst_range(my_lst)
    print("Range is",r)

main()
Debugging demo: 3 types of bugs

1) Syntax error

2) Error message

3) No errors but your program is not doing what you want
Takeaways from min/max/range

1) **Write a function, test the function!** THEN move on

2) Testing of each separate function can be done in main or the shell

3) Use **print** A LOT when you are testing and debugging your functions

4) Three types of errors
   A) **Syntax** -> look on the previous line or reindent your code
   B) **Error message** -> follow the line numbers from the bottom up
   C) **No errors but not doing what you expect** -> don’t panic! Start printing!
Go over Lab 3
Lab 3, Part D Solution

# CSC 111, Lab 3, Part D Solution
# Author: Sara Mathieson
# A program to generate random strings of DNA

import random

def main():
    bases = ["A","C","G","T"]

    num_genes = eval(input("Enter the number of genes: "))
    num_bases = eval(input("Enter the number of bases in each gene: "))
    print()

    for i in range(num_genes):
        gene = ""  # Initialize a variable outside the loop.
        for j in range(num_bases):
            b = random.randint(0,len(bases)-1)
            gene = gene + bases[b]
            print("gene" + str(i) + ": " + gene)

    main()
Lab 3, Part E Solution

# CSC 111, Lab 3, Part E Solution
# Author: Sara Mathieson
# A program compare two DNA sequences

import random

def main():
    human = "ATA?CAAGACCTCGTATTGAATACGCGCCATGTGAGTATCCTATC?GA"
    chimp = "ATAACAAGAGCTAGTTATTA?TACTGCGCCATGTGAGAAATCCTATAGGA"

    n = len(human)
    diff = 0
    same = 0
    unknown = 0

    for i in range(n):
        if human[i] == "?" or chimp[i] == "?:"
            unknown = unknown + 1
        elif human[i] != chimp[i]:
            diff = diff + 1
        else:
            same = same + 1

    print(same, "bases are the same.")
    print(diff, "bases are different.")
    print(unknown, "bases: either human or chimp or both unknown.")

main()
Reading and Writing Files
Reading and writing files: why now?

- Working with files involves using both string and list methods

  \[17\]
  \[10\]
  \[1\]
  \[12\]
  \[5\]
  \[18\]
  \[15\]
  \[16\]
  \[6\]
  \[14\]
Reading and writing files: why now?

Working with files involves using both string and list methods.

Extend our idea of sequences:
A) A string is a sequence of characters
B) A list is a sequence of elements
C) A file is a sequence of lines
Reading and writing files: why now?

- Working with files involves using both string and list methods.

- Extend our idea of sequences:
  A) A string is a sequence of characters
  B) A list is a sequence of elements
  C) A file is a sequence of lines

- We can start to answer real questions using large datasets.
File writing demo
Important file methods

+ **open**: i.e. `my_file = open(“numbers.txt”, “r”)`
  
  Note: built-in method, 3 modes for now: read (“r”), write (“w”), append(“a”)

+ **read**: i.e. `all_text = my_file.read()`
  
  Note: returns the entire file as a single (potentially large) string

+ **readline**: i.e. `line = my_file.readline()`
  
  Note: returns the next line of the file

+ **readlines**: i.e. `all_lines = my_file.readlines()`
  
  Note: returns the entire file as a list of lines

+ **close**: i.e. `my_file.close()`
  
  Note: should always be done after reading or writing a file
Two 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]`