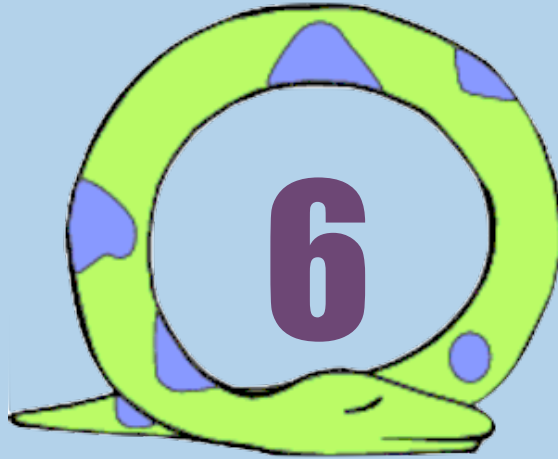


CS116 TUTORIAL



ITERATION

CLICKER QUESTION 1

When is the CS116 midterm?

- A. It was this Tuesday.
- B. Monday, March 2nd at 7 AM.
- C. Do we have a midterm?
- D. Monday, March 2nd at 7 PM.

REMINDERS

- Midterm Q&A session is TOMORROW!!!!!! (2:00-3:50 at STCI012)
- Midterm is on March 2nd
- Check A5 solution for preparing for the Midterm!
- Assignment 6 is due at 10 AM on Wednesday, March 11th
 - Hint: Best time to come seek help is any day that's not the day before the due time. (Less competition!)

TODAY → **LOOPS!!**

- 2 Types of loops
 - `while`
 - `for`
- Nested Loops

REVIEW – WHILE LOOPS

```
***initialize variables***
```

```
while condition:
```

```
    ***body of while, including***
```

```
    ***update of variables***
```

This part will
continuously be
executed until
condition is
False

- The body of the while loop will execute until `condition == False`
- The `condition` is only checked before each execution of the loop body.
- Variables **MUST** be updated, otherwise there might be an infinite loop!

(Sort of like `maximum recursion depth`)

REVIEW – FOR LOOPS

```
for item in collection:  
    *** body of loop ***
```

A collection can be like something like a list, a string, etc.

- The body of the for loop will execute `len(collection)` times, once for every element in collection
- Similar to `map`; goes through every element in the collection

CLICKER QUESTION 2

What is the value of L after going through this for loop?

```
L = [0, 1, 2, 3, 4, 5, 6]
```

```
for x in L:
```

```
    L[(x+1)%len(L)] = x
```

A. [6, 0, 1, 2, 3, 4, 5]

B. [6, 0, 2, 2, 4, 4, 6]

CLICKER QUESTION 2

`L = [0, 1, 2, 3, 4, 5, 6]`

- 1) For `x = 0`, `L[1] = 0`, `L = [0, 0, 2, 3, 4, 5, 6]`
- 2) For `x = 0`, `L[1] = 0`, `L = [0, 0, 2, 3, 4, 5, 6]`
- 3) For `x = 2`, `L[3] = 2`, `L = [0, 0, 2, 2, 4, 5, 6]`
- 4) For `x = 2`, `L[3] = 2`, `L = [0, 0, 2, 2, 4, 5, 6]`
- 5) For `x = 4`, `L[5] = 4`, `L = [0, 0, 2, 2, 4, 4, 6]`
- 6) For `x = 4`, `L[5] = 4`, `L = [0, 0, 2, 2, 4, 4, 6]`
- 7) For `x = 6`, `L[0] = 6`, `L = [6, 0, 2, 2, 4, 4, 6]`

- Be careful of mutating your `collection` inside the loop!
 - Never change the length of the same collection that you are iterating over in a for loop

WHILE LOOP VERSION OF A FOR LOOP

```
for item in collection:  
    ***body of loop***
```

```
i = 0  
while i < len(collection):  
    item = collection[i]  
    ***body of loop*** (same as above)  
    i = i + 1
```

REVIEW – NESTED LOOPS

```
for i in collection1:
```

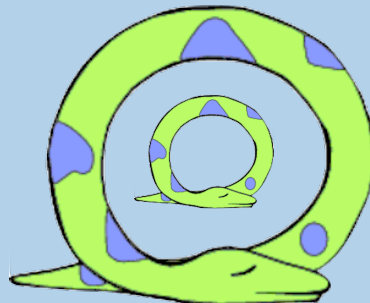
```
    *** body of outer for ***
```

```
        for j in collection2:
```

```
            *** body of inner for ***
```

Body of
outer
for
loop

- For each `i` in `collection1`, the inner for loop will be executed
- Examples of possible `collection1`:
 - list of nested lists
 - lists of strings



The inner for loop will be executed `len(collection1)` times.

The body of the inner for will execute `len(collection2)` times for each value of `i`.

CLICKER QUESTION 3

What is L[0] after calling A(L)?

A. 0

B. 3

C. Error

D. None of the above

```
L = [0, 1, 2, 3]
```

```
def A(lst):  
    m = lst[0]  
    for n in lst:  
        if n > m:  
            m = n  
            n += 1  
    return m
```

WHAT SHOULD MY LOOP COUNTER BE?

Examples for some *meaningful* counter names:

- `i to n => integer`
- `L => List`
- `s => string`
- `c => characters (strings of length 1)`
 - *You are always allowed to use other meaningful names*

`i, j, k` convention for integer counters are in fact inherited from **Fortran**. In Fortran, integer variables had to start with the letters `i` through `n`.

- *This is only for interest, materials on this will not be tested on exam.*

QUESTION 1 - ALL_SAME_TYPE

Write a function `all_same_type` that consumes a list, called `lst`, and returns `True` if all members of that list are of the same type, else `False`.

For example:

```
all_same_type([2, 5, 3]) => True
all_same_type([2, 'R', 4.56]) => False
```

Note that Python's built-in `type` function does not distinguish between types of lists:

```
i.e. type([1,2]) == type(['a', 'b'])
```

QUESTION 2 – MAX_EVEN_SUM

Write a Python function `max_even_sum` that consumes a **nonempty list, `lst`**. Each value in `lst` is a list of positive integers. It computes the sum of the even integers in each of the element lists in `lst`, and returns the largest out of these sums.

If an element list contains no even integers, its sum is zero.

For example:

```
max_even_sum([[ ], [3], [2,4,6]]) => 12
```

QUESTION 3 – SUM_DIGITS

Write a Python function `sum_digits` **using loops** that consumes a Nat (called `n`), and returns a number represents the summation of its digits.

Examples:

```
sum_digits(1) => 1
```

```
sum_digits(55) => 10
```

```
5+5 = 10
```

QUESTION 4 – MAKE_LIST

Write a Python function `make_list` that consumes a natural number `n` and returns a list of strings. The produced list will look like

```
["", "1", "22", "333", "4444", "55555", ... ,  
 "nnnnn...nnnn"]
```

where the last element is the number `n` repeated `n` times.

For example:

```
make_list(0) => [""]
```

```
make_list(3) => ["", "1", "22", "333"]
```


QUESTION 5 – VALID_INPUT

Write a function called `valid_input` that consumes a **string to be used as the prompt**, `prompt`, **a list of strings of valid inputs**, `valid`, and a positive integer `max_guess`.

The function should continuously prompt the user for input until the user enters a value in the list `valid`, and then return that value, or print a message when maximum number of guess is reached. If the user enters an invalid value, the function will let them know by printing: "Invalid input. Try again." to the screen. If maximum number of guess is reached, the function will print "Maximum number of guesses reached" and return `None` in this case.

QUESTION 5 (CONTINUED)

For example:

If the user enters "6", "5", and "3",

```
valid_input("Enter a digit < 5: ",  
           ["0", "1", "2", "3", "4"], 5) => "3"
```

and the following is printed:

```
Enter a digit < 5: 5
```

```
Invalid input. Try again.
```

```
Enter a digit < 5: 3
```

Note: You may assume that the user enters input that is the correct type.