REMINDER

• Assignment 06 due next Wednesday, June 28th, 10AM
TODAY

• Loops!
  – while
  – for
  – nested
**REVIEW – WHILE LOOPS**

***initialize variables***

```python
while condition:
    ***body of while, including***
    ***update variables***
```

- It will continue to execute the body of the `while` loop until `condition == False`
- Variables **MUST** be updated, otherwise there might be an infinite loop!
for item in collection:
    *** body of loop ***

- It will execute the body of the for loop len(collection) times, once for every element in collection
- Similar to map; goes through every element in the collection
- Be careful of mutating your collection inside the loop!
- Don’t alter the length of collection inside the body
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**

```python
for i in collection1:
    *** body of outer for ***
    for j in collection2:
        *** body of inner for ***
```

- For each `i` in `collection1`, the inner `for` loop will be executed `len(collection1)` times.
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:

```python
type([1,2]) == type(['a','b']) => True
```
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:

```python
max_even_sum([[], [3], [2,4,6]]) => 12
```
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"]`
Write a function called `valid_input` that consumes a string to be used as the 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 guess is reached".

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.
WHAT SHOULD MY LOOP COUNTER BE?

Examples for some meaningful counter names:

- *i to n* => integer
- *l* => list
- *s* => string
  - 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.