# Loops
# Tuples
# Named Tuples

# Announcements

- H07 due next Wed (11/01)
- Midterm stats:
- Average 86.54
- Medium: 89.75
- std: 11.91

Common mistakes on the midterm:

https://ucsb-cs8-f18.github.io/lectures/lect08/

# Loops:
# Repetition without being repetitive

Syntax

```
for item in some_collection:
    # Code
```

Example: Iterating through collections

- Print each character of a string
- Print each element of a list
- Print each element of a tuple

# Concept Test

- What is the output of this code?

```
for x in [1, 2, 3]:
    print('Hello'*x) # using x inside the loop
```

A. 1 2 3

B. 'Hello' is printed 3 times

C.  Hello

   HelloHello

   HelloHelloHello

D. None of the above

# Range( ) function

- Used in a loop when we know the number of times we want to repeat executing some code

```
range(5)      # think of it as producing a list [0, 1, 2, 3, 4]
range(1, 5)   # The first parameter is a starting value
              # The second parameter is the stopping value
              # Output [1, 2, 3, 4]
range(0, 10, 2)  # The third parameter is the step count
              #[0, 2, 4, 6, 8]


for x in range(5):
    print('Hello')
```

# Concept Test

What is the output of this code?

```
for x in range(1,4,2):
    print(2**x, end = " ")
```

A. 1 4 2

B. 2 16 4

C. 2 8 16

D. 2 8

E. None of the above

# Loops and conditionals

```
def containsOddNumber(lst):
   "return True if any element in lst is odd,
   otherwise return False"
```

```
def isListOfNumbers(lst):
   "return True if all the elements in lst are
   numbers, otherwise return False"
```

# Concept Test

```
def containsOddNumber(lst):
    '''return True if any element in lst is
    odd, otherwise return False'''
    for x in lst:
        if (x % 2 == 1):
            return True
        else
            return False
```

Is the above implementation correct?(Why or Why not)

A. Yes

B. No

# The accumulator pattern

Useful for "accumulating" something while going through a collection.

Example: Count the number of times, count the number of characters in a string, ...

```
def countElements(lst):
   "returns the number of elements in lst"


def countOddNumbers(lst):
   "returns the number of odd numbers in lst"


def count(name, letter):
   "return the number of times letter appears in
    name in either upper or lower case"
```

# More on the accumulator pattern

```
def countWords(sentence):
   "returns the number of words in the sentence"


def countWords(sentence, len):
   "returns the number of words in the
   sentence with length greater than len"
```

# Tuples

- Similar to lists: store a sequence of elements

lst = [ 10, 20] //ex of a list

tup = (10, 20) //ex of a tuple

- Elements are ordered an can be accessed using the appropriate index

tup[0]

tup[1]

- Different from lists in the following ways
  - Can't change an element in the tuple
  - Can't sort the elements in a tuple

# Named Tuples

- Used to package data with multiple attributes: e.g. representing a student in your program
- A student's attributes may be: name, perm number, major etc.
- Named tuples make it easier to access each attribute

```python
from collections import namedtuple

#Design your named tuple object
Student = namedtuple('Student', 'name perm major gpa')

# Create objects of type Student
s1 = Student("Jack", 123443, CS, 3.8)
s2 = Student("Mary", 8932737, CE, 3.9)

# Access the elements of the objects
print(s1.name, s1.perm)
```