Tuples
Named Tuples
Accumulator pattern
Nested Loops

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

```
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)
```

The accumulator pattern: ex01

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”
```

The accumulator pattern: ex02

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

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

Accumulator pattern: ex03

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

Accumulator pattern: ex04

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

The accumulator pattern: ex05

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

```
def createListOfOdd(lst):
  "returns a new list that contains all the odd
  numbers in lst”
```

Nested Loops

```
def drawRectangle(width, height):
  ”print a rectangle with given width
  and height using the character *
  (instead of turtle)”
```

```
  For example drawRectangle(5,3)
  should print

  *****
  *****
  *****
```

Nested Loops

```
def drawTriangle(height):
  ”print a right triagle with given
  height using stars(*). Assume the
  size of the base and height are
  equal”

  For example drawTriangle(3)
  should print

  *
  **
  ***
```