

# Tuples

# Tuple

- Similar to a list except it is immutable.
- Syntactically, it is comma-separated list of values.
  - `tuple = 'a', 'b', 'c', 'd', 'e'`
- To create a tuple with a single element, we have to include the comma:
  - `t1 = ('a',)`
  - `type(t1)`  
`<type 'tuple'>`
- Without the comma, Python treats ('a') as a string in parentheses:
  - `t2 = ('a')`
  - `type(t2)`      ??

- The operations on tuples are the same as the operations on lists
- The index operator selects an element from a tuple.
  - `tuple = ('a', 'b', 'c', 'd', 'e')`
  - `tuple[0]`
  - `'a'`
- The slice operator selects a range of elements.
  - `tuple[1:3]`
  - `('b', 'c')`
- But if we try to modify one of the elements of the tuple, we get an error.
  - `tuple[0] = 'A'`
  - `TypeError: object doesn't support item assignment`

# Example

- we can't modify the elements of a tuple, we can replace it with a different tuple:
  - `tuple = ('A',) + tuple[1:]`
  - `tuple`
  - `('A', 'b', 'c', 'd', 'e')`

# Tuple Assignmmt

- To swap the values, using conventional assignment statements, we have to use the assignment statement with a temporary variable.
  - `temp = a`
  - `a = b`
  - `b = temp`
- Python provides a form of tuple assignment that solves this problem neatly:
  - `a, b = b, a`
- The number of variables on the left and the number of values on the right have to be the same.

# Tuples as Return Values

- Functions can return tuples as return values.
- For example:
  - `def swap(x, y):`  
    `return y, x`
- Then we can assign the return value to a tuple with two variables:
  - `a, b = swap(a, b)`

```
def swap(x, y): # incorrect version
```

```
    x, y = y, x
```

- If we call this function like this:

```
swap(a, b)
```

- a and x are aliases for the same value. Changing x inside swap makes x refer to a different value, but it has no effect on a in main . Similarly, changing y has no effect on b.

# Random Numbers

- The random module contains a function called random that returns a floating point number between 0.0 and 1.0.
- Each time you call random, you get the next number in a long series.

```
from random import *
```

```
for i in range(10):
```

```
    x=random()
```

```
    print x
```



# List of Random Numbers

- ```
def randomList(n):  
    s = [0] * n  
    for i in range(n):  
        s[i] = random()  
    return s
```

Output:

randomList(8)

0.15156642489

0.498048560109

0.810894847068

0.360371157682

0.275119183077

0.328578797631

0.759199803101

0.800367163582

# Counting

- Divide the problem into sub problems and look for sub problems that fit a computational pattern
- Traverse a list of numbers and count the number of times a value falls in a given range.
- Eg:

```
def inBucket(t, low, high):  
    count = 0  
    for num in t:  
        if low < num < high:  
            count = count + 1  
    return count
```

This development plan is known as pattern matching.

# Many Buckets

- With two buckets, range will be:
  - low = `inBucket(a, 0.0, 0.5)`
  - high = `inBucket(a, 0.5, 1)`
- But with four buckets it is:
  - bucket1 = `inBucket(a, 0.0, 0.25)`
  - bucket2 = `inBucket(a, 0.25, 0.5)`
  - bucket3 = `inBucket(a, 0.5, 0.75)`
  - bucket4 = `inBucket(a, 0.75, 1.0)`

- If the number of buckets is numBuckets, then the width of each bucket is  $1.0 / \text{numBuckets}$ .

```
numBuckets = 8
```

```
buckets = [0] * numBuckets
```

```
bucketWidth = 1.0 / numBuckets
```

```
for i in range(numBuckets):
```

```
    low = i * bucketWidth
```

```
    high = low + bucketWidth
```

```
    buckets[i] = inBucket(t, low, high)
```

```
print buckets
```

- `Len(tuple)`
- `Min(tuple)`
- `Max(tuple)`
- `Cmp(tuple1, tuple2)`
- `Tuple(list)`

# Questions

- Q1. Create a tuple and try to changing value of any one element, also display the length of list.
- Q2. Create a tuple having single element and append two tuples.
- Q3. Create a tuple and sort it.
- Q4. Create a tuple of numbers and print sum of all the elements.
- Q5. Program to compare elements of tuples.
- Q6. Program to find maximum and minimum of tuple.
- Q7. Count the occurrence of element in tuple in a specific range.
- Q8. Reverse a tuple.
- Q9. Write a loop that traverses the previous tuple and prints the length of each element. What happens if you send an floating number in index?