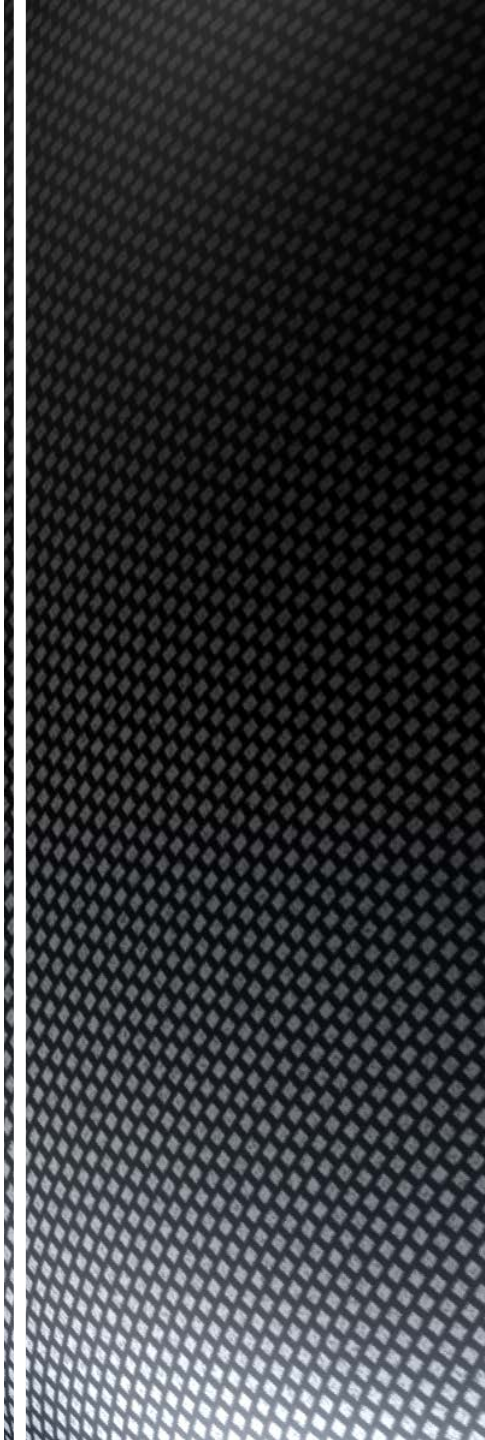
COMPSCI 107
Computer Science Fundamentals

Lecture 05 – Dictionaries



Exercise

- Consider the following methods of sorting information in a sequence. What is the output produced by each program?

```
x = [4, 2, 3, 1]
y = x
z = sorted(x)
print(x, y, z)
```

```
x = [4, 2, 3, 1]
y = x
z = x.sort()
print(x, y, z)
```

Dictionary

- An unordered set of ***key : value*** pairs
  - keys must be unique (within a given dictionary)
  - keys must be *immutable*

```
my_dict = { 'alux001' : 'Andrew',
            'afer023' : 'Adriana',
            'rshe001' : 'Robert' }
```

```
>>> my_dict['alux001']
'Andrew'
>>> my_dict['rshe001']
'Robert'
>>> my_dict['afer023']
'Adriana'
```

Adding entries to a dictionary

- Variables are created when you assign a value for the first time
  - Assigning a value to a new dictionary entry creates that entry
  - Assigning a value to an existing dictionary entry changes that entry

```
my_dict = {}
my_dict['alux001'] = 'Andrew'
my_dict['rshe001'] = 'Robert'
my_dict['afer023'] = 'Adriana'

my_dict['alux001'] = 'Andrew Luxton-Reilly'
```

Deleting an element from a dictionary

- Use the **del** operator

```
del my_dict['alux001']
```

- Use the **pop** method

```
my_dict.pop('alux001')
```

Iterating through the dictionary

- Order of items in a dictionary is undefined

- Can get a "view" of the keys, values, or (key, value) pairs
  - Can iterate through the "view"

```
for k in my_dict.keys():
    print(k)
```

```
for v in my_dict.values():
    print(v)
```

```
for i in my_dict.items():
    print(i)
```

- Views are dynamic, so they change when the dictionary changes

Getting a value from the dictionary

- Get a value associated with a given key
  - Use the index, use get, or use get with default value

```
>>> my_dict['alux001']
'Andrew'
>>> my_dict['ALUX999']
KeyError: 'ALUX999'
```

```
>>> my_dict.get('alux001')
'Andrew'
>>> my_dict.get('ALUX999')
>>>
```

```
>>> my_dict.get('alux001', 'Not valid')
'Andrew'
>>> my_dict.get('ALUX999', 'Not valid')
'Not valid'
```

Exercise

- Write a function that accepts a string as an argument and prints out a frequency table of all the characters in the string, along with their frequency. The table should be sorted in order of the characters.

```
>>> frequency('this is a short sentence')
  4
a 1
c 1
e 3
h 2
i 2
n 2
o 1
r 1
s 4
t 3
```

Tuples

- A tuple is a sequence, much like a list
  - A tuple is *immutable*
  - Enclosed in brackets (x, y)

```
>>> point = (1, 2)
>>> point
(1, 2)
>>> point[0]
1
>>> point + (3, 4)
(1, 2, 3, 4)
```

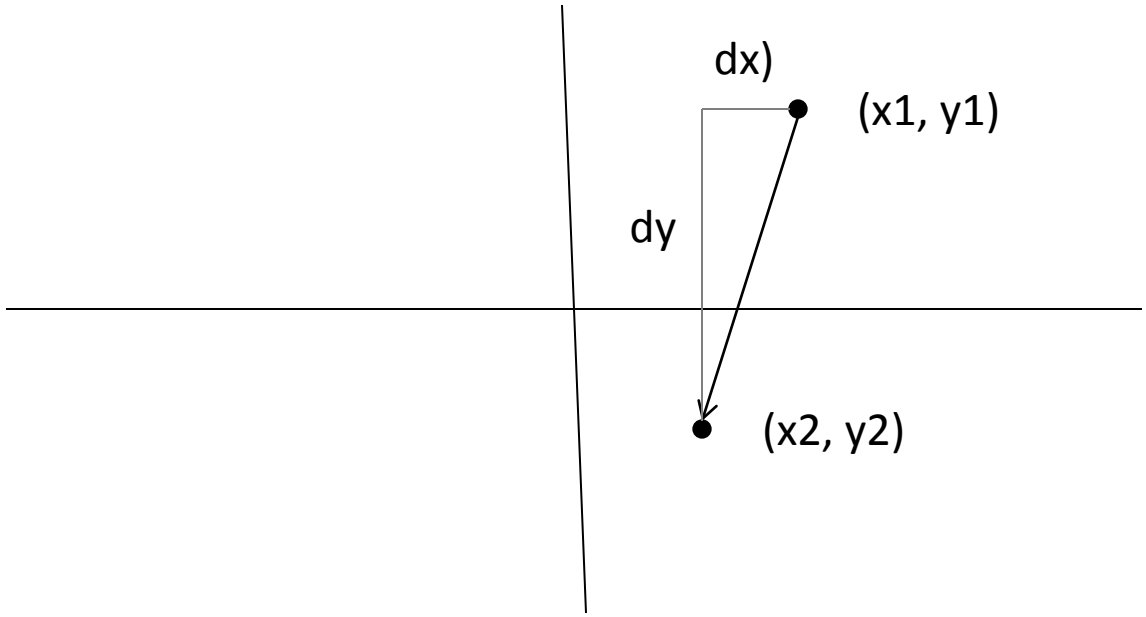
Returning multiple values

```
def translate(point, vector):
    """Translate a point by a given vector

    Arguments:
        point – a tuple representing an (x, y) point on
            the cartesian plane
        vector – a tuple representing a translation of
            (dx, dy) on the cartesian plane

    >>> translate((0, 0), (4, 5))
    (4, 5)
    """
    x = point[0] + vector[0]
    y = point[1] + vector[1]
    return (x, y)
```

```
>>> new_x, new_y = translate((1, 2), (3, 4))
```

Exercise: Difference between two points

- Write a function that calculates the difference between two points and returns a vector (dx, dy) to translate from one to the other.



Example: Sparse Matrix

- A sparse matrix is a matrix with few non-zero values.

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 7 & 0 & 0 & 0 & 4 \\ 0 & 0 & 0 & 0 & 0 & 2 & 0 \end{bmatrix}$$

- The most obvious way to represent the matrix is a list of lists:

```
matrix = [ [0, 0, 0, 0, 0, 0, 0],
           [0, 1, 0, 0, 0, 0, 0],
           [0, 0, 0, 3, 0, 0, 0],
           [0, 0, 7, 0, 0, 0, 4],
           [0, 0, 0, 0, 0, 2, 0] ]
```

Example: Sparse matrix

- Alternative implementation

- Use a dictionary
  - Use the location as the key
  - Store location as a tuple (x, y)

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 7 & 0 & 0 & 0 & 4 \\ 0 & 0 & 0 & 0 & 0 & 2 & 0 \end{bmatrix}$$

```
matrix = {(1, 1) : 1,
          (2, 3) : 7,
          (3, 2) : 3,
          (5, 4) : 2,
          (6, 3) : 4 }
```

Accessing the sparse matrix

```
matrix = {(1, 1) : 1,
          (2, 3) : 7,
          (3, 2) : 3,
          (5, 4) : 2,
          (6, 3) : 4 }
```

- Use the location (x, y) as the index

```
>>> matrix[(3, 2)]
3
>>> matrix[(0, 0)]
KeyError: (0, 0)
```

- Use the location (x, y) as the index, but use a default value

```
>>> matrix.get((4, 3), 0)
0
```

Summary

- Dictionaries are also known as associative arrays or hash tables
  - Consist of key : value pairs
  - keys must be immutable
  - Syntax is {a : b, c : d, …}
- Adding an item
  - d[key] = value
- Deleting an item
  - d.pop(key)
- Getting an item
  - d.get(key, default)