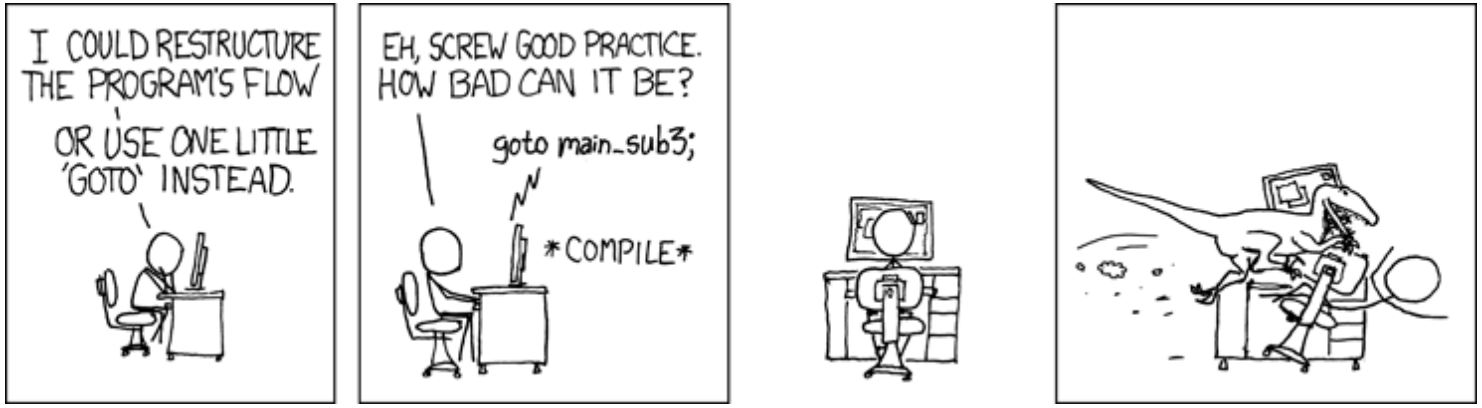
CSc 110, Spring 2018

Lecture 34: 2D Structures

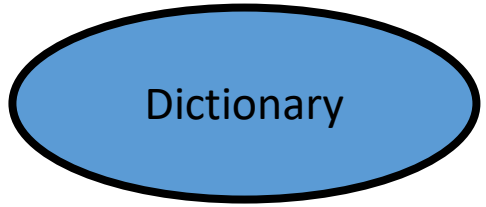
Adapted from slides by Marty Stepp and Stuart Reges



Using dictionaries

- A dictionary allows you to get from one half of a pair to the other.
  - Remembers one piece of information about every index (key).

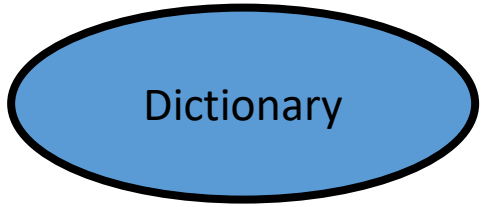
```
#          key        value
my_dict["Suzy"] = "206-685-2181"
```



  - Later, we can supply only the key and get back the related value:

    Allows us to ask: *What is Suzy's phone number?*

```
my_dict["Suzy"]
```




```
"206-685-2181"
```

Dictionaries and tallying

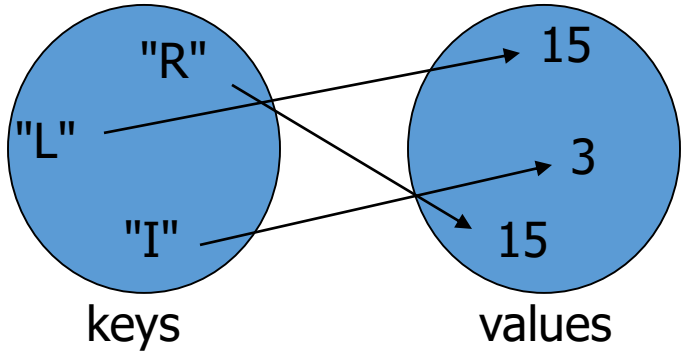
- a dictionary can be thought of as generalization of a tallying list
  - the "index" (key) doesn't have to be an `int`
  - count digits: `2209231090`7

index	0	1	2	3	4	5	6	7	8	9
value	3	1	3	0	0	0	0	1	0	2

```
# (Roosevelt), (L)andon, (I)ndependent
```

  - count votes: `"RLLLLLLRRRRRLLLLLLRLRRIRLRRIRLLRIR"`

key	"R"	"L"	"I"
value	15	15	3



Dictionary operations

`items()`	return a new view of the dictionary's items ((key, value) pairs)
`pop(`**`key`**`)`	removes any existing mapping for the given key and returns it (error if key not found)
`popitem()`	removes and returns an arbitrary (key, value) pair (error if empty)
`keys()`	returns the dictionary's keys
`values()`	returns the dictionary's values

You can also use `in`, `len()`, etc.

Looping over a set or dictionary?

- You must use a `for element in` structure loop
  - needed because sets have no indexes; can't `get` element `i`

Example:

```
for item in a:
    print(item)
```

Outputs:

```
the
happy
hello
```

items, keys and values

- `items` function returns tuples of each key-value pair
  - can loop over the keys in a for loop

```
ages = {}
ages["Merlin"] = 4
ages["Chester"] = 2
ages["Purrcival"] = 12
for cat, age in ages.items():
    print(cat + " -> " + str(age))
```

- `values` function returns all values in the dictionary
  - no easy way to get from a value to its associated key(s)
- `keys` function returns all keys in the dictionary

Exercise

- Use word counts to figure out if a document is positive or negative
  - Count all of the positive words and count all of the negative words.
  - Whichever count is bigger is the sentiment of the document.

- How do we know which words are positive and which are negative?

Exercise

Consider the following function:

```
def mystery(list1, list2):
    result = {}
    for i in range(0, len(list1)):
        result[list1[i]] = list2[i]
        result[list2[i]] = list1[i]
    return result
```

What is returned after calls with the following parameters?

`list1: [b, l, u, e]` `list2:[s, p, o, t]`

dictionary returned:_____

`list1:[k, e, e, p]` `list2: [s, a, f, e]`

dictionary returned:_____

`list1:[s, o, b, e, r]` `list2: [b, o, o, k, s]`

dictionary returned:_____

What is the right structure?

- You want to store a bunch of colors so you can later choose one at random.
- Batting order of a baseball team.
- Students names and their grades on a project.
- Friends names and their phone numbers
- Height, width and location of a sports field.
- Movies a person has watched.
- Items in a shopping cart.
- A student's grades.

What is the right structure?

- The grades for all students in a class
- All books in a store arranged by category
- Many recipes each containing many steps
- Phone numbers that have been called this month on a phone plan divided by area and country code for billing simplicity

Exercise

- We would like to store data for the class so that we can:
  - Access the entire class list easily
  - Access a section list easily

- What structure is appropriate for this problem?
  - Sometimes it can be helpful to store a structure inside another structure