# Intro to Computer Science

**Previous**

- Boolean algebra
- while-loops

**Next**

- Dictionaries

| Readings | |
|---|---|
| Gaddis | • Chapter 3.5, 4.2 |

| Readings | |
|---|---|
| Gaddis | • Chapter 9.1 |

# Dictionaries

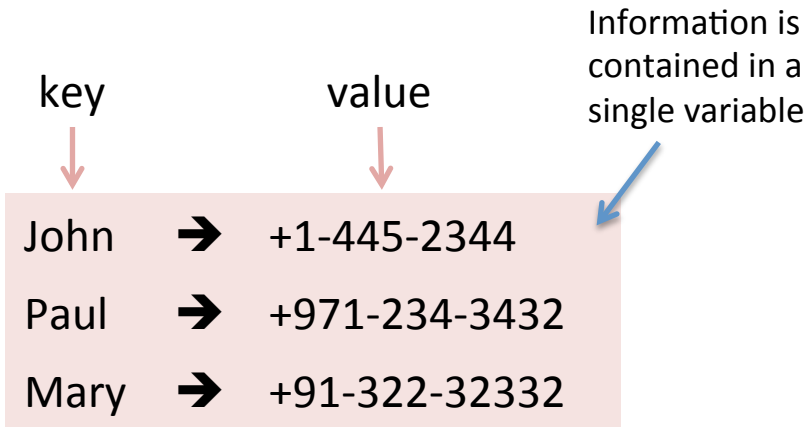
- Python has a data type specifically designed for such key-value pairings
  - Known as a *dictionary*
- Most other modern languages have a similar concept
  - hash table, associative array, map, symbol table, etc.

# There's a type for that!

| To us | To Python | Examples |
|---|---|---|
| Integers | int | …, -2, -1, 0, 1, 2, … |
| Real numbers | float | 12.345, 3.14 |
| Words | str | 'Hello', 'World' |
| Lists | list | [ 3, 2.1, 'blast off!' ] |
| Boolean values | bool | True, False |
| Dictionaries | dict | { 'key1': 12, 'key2': 10 } |

# Dictionaries

- A dictionary is essentially a mapping between *keys* and *values*
  - Given a key, returns a particular value
- Allows us to *associate* a name with a value
  - Lists force us to think in terms of
    - order: where in the list are we
    - indices: how to obtain the value there
  - Much more intuitive
- Opens up new potential for abstraction



# Manipulating dictionaries

(Or, a road map for the next few minutes)

| Topic | Operations (sub-topics) |
|---|---|
| 1. Creation | • Static creation<br>• Dynamic creation |
| 2. Alteration | • Add an item<br>• Remove and item |
| 3. Iteration | • Over keys<br>• Over values |
| 4. Existence | • `in`<br>• `not in` |

# Using a dictionary

```
dtc = {
    'John': 1232233222,
    'Paul': 1437232452,
    'Mary': 3566244723
}

print(dtc['John'])

dtc['Richard'] = 3128923863

dtc['Paul'] = 4228925667

del dtc['John']

dtc = {}
```

Create a new directory

Access John's phone number

Add phone number for Richard

Update Paul's phone number

Delete John's number

Create the empty directory

# Creation

```
1
  dtc = {
      'John': 1232233222,
      'Paul': 1437232452,
      'Mary': 3566244723
  }
```

```
2
  dtc['Richard'] = 3128923863

  dtc['Paul'] = 4228925667
```

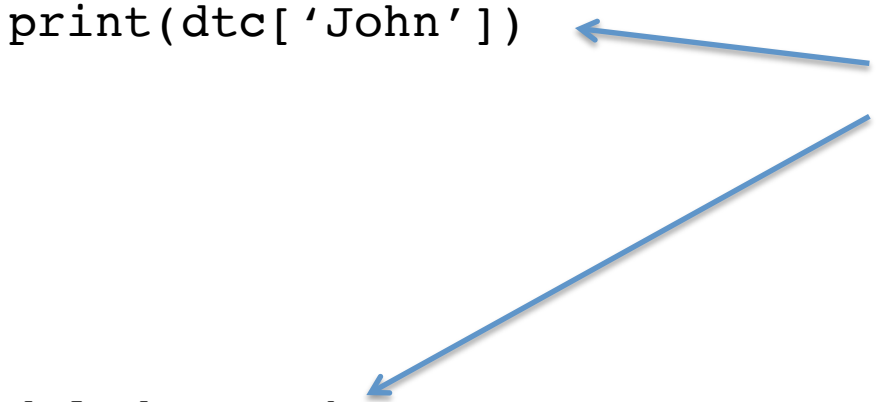
```
3
  dtc = {}
```

**Three methods of creation**

1. Explicit naming of key-value pairs
   - Note the braces, the colon, and the comma!
2. Assignment using subscript notation
3. Empty dictionary initialization
   - Akin to [] when creating lists

# Access

```
print(dtc['John'])
```



```
del dtc['John']
```

- Access refers to obtaining a value from the dictionary
  - This is different from assignment
- Uses subscript notation
  - Like lists, but not just integers
- When accessing an element the key must exist!

# Are you in?

- Accessing a key that is not in a dictionary raises an error in Python
  - Like accessing an index that does not exist in a list
- To help, Python has the `in` operator

These are Boolean expressions

| | |
|---|---|
| key in dictionary | `if 'John' in dtc:`<br>`  print(dir['John'])` |
| key not in dictionary | `if 'John' not in dtc:`<br>`  print('Nope')` |

- This is the same 'in' that's used in for-loops, but because the *context* is different, so too are the semantics

# Iterating over a dictionary

**Over keys**

- Use the ‘keys’ method

```
dtc = {
    ‘Bob’: 89734987,
    ‘Mary’: 23873243
}

for i in dtc:
    print(i, dtc[i])

for i in dtc.keys():
    print(i, dtc[i])
```

**Over values**

- Use the ‘values’ method

```
dtc = {
    ‘Bob’: 89734987,
    ‘Mary’: 23873243
}

for i in dtc.values():
    print(i)
```