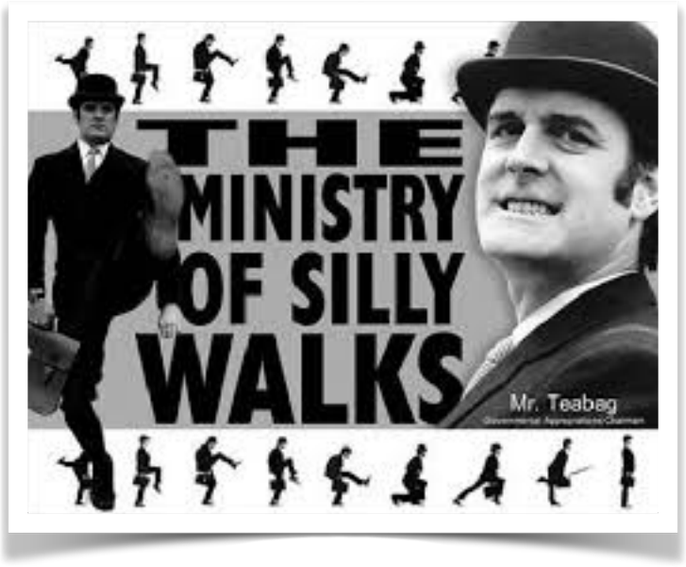
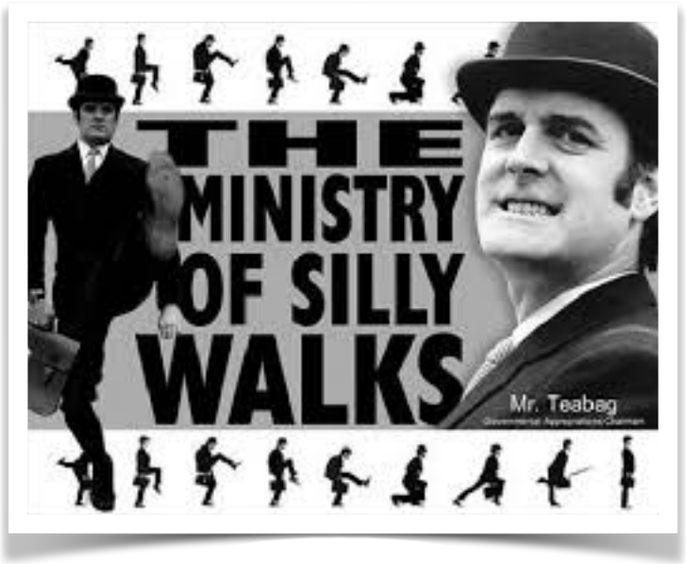
Dictionaries

Thomas Schwarz, SJ



Dictionaries

Python



Dictionaries

- Python has a efficient association data structure — the dictionary
  - Dictionary pairs keys with values
  - Useful for: indices
  - Useful for: translations
  - Useful for: quick lookups
    - E.g.: first letters —> full email address
    - E.g.: human-readable URL —> IP address
    - …

Dictionaries

- Dictionaries are key-value stores
  - Keys — anything, but needs to be immutable
    - Remember: Lists are mutable, strings are immutable
  - Value — anything

Dictionaries

- Dictionaries are created by using curly brackets
  - Can use lists

```
dicc = {1: 'eek', 2: 'do', 3: 'teen'}
```

  - Or can use assignment

```
dicc = {}
   dicc[1] = "uno"
   dicc[2] = "dos"
   dicc[3] = "tres"
```

- Values are assigned / retrieved using the bracket notation

Dictionary

- Dictionary `dicc={}`
- Accessing values:

```
dicc['key']
```

```
>>> dicc = {1: "uno", 2: "dos", 3: "tres"}
>>> dicc[1]
'uno'
>>> dicc[1] = "one"
>>> dicc[1]
'one'
```

  - With default value

```
dicc.get(key, default_value)
```

  - Or with if - else

```
if key in dicc:  value = dicc[key]
```

- Creating / changing values

```
dicc['key'] = value
```

Dictionary

- Deleting from a dictionary

```
dicc = {}
```

  - Use the del keyword
    - Raises a key error if the key is not in the dictionary

```
if key in dicc:
    del dicc[key]
```

  - Use the pop method, which returns the value

```
value = dicc.pop(key)
value = dicc.pop(key, default)
```