LECTURE 11: DICTIONARIES

Hung-Yu Wei

Dictionary

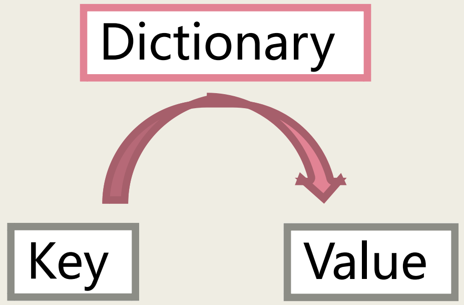
- Mapping of key-value pair

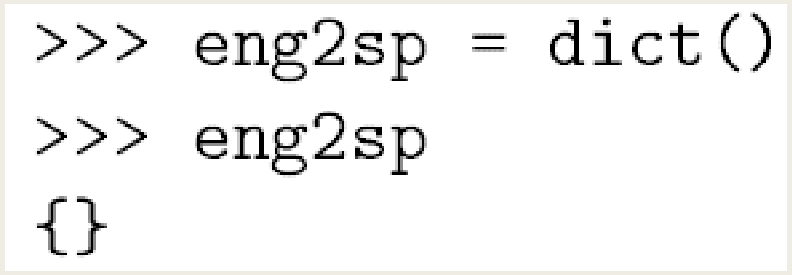
  value = dictionary (key)

- Example
  - *English to Spanish dictionary*

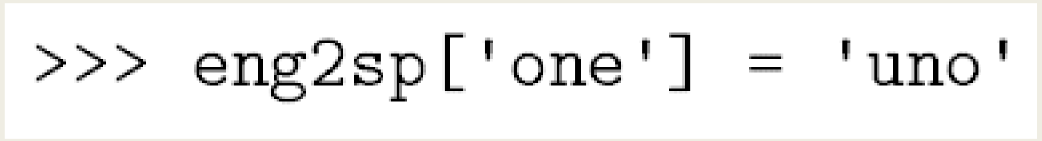
- Syntax

  dict()

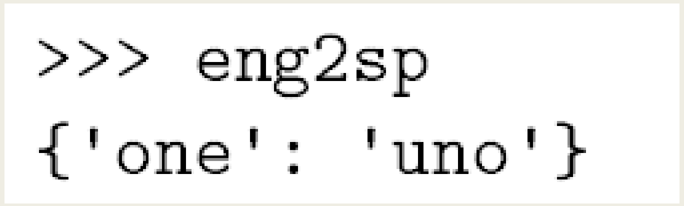




```
>>> eng2sp = dict()
>>> eng2sp
{}
```



```
>>> eng2sp['one'] = 'uno'
```



```
>>> eng2sp
{'one': 'uno'}
```

Key-value pairs

- Order of key-value pairs
  - *Not printed in order (actually unpredictable)*
  - *The order is not important*
    - The important thing is the mapping relationship

```
>>> eng2sp = {'one': 'uno', 'two': 'dos', 'three': 'tres'}
```

```
>>> eng2sp
{'one': 'uno', 'three': 'tres', 'two': 'dos'}
```

Operations

- []
  - *Look up with a key*
  - ***[****key****]***
- Error
  - *Cannot find a key in the dictionary*
  - ***KeyError***
- len()
  - *Length of a dictionary*
  - ***len****(dictionary)*

```
>>> eng2sp['two']
'dos'
```

```
>>> eng2sp['four']
KeyError: 'four'
```

```
>>> len(eng2sp)
3
```

Finding: if key or value exist

- **in**
  - *Whether a key is in a dictionary*
  - *key* ***in*** *dictionary*
- **values()**
  - *Whether a value is in a dictionary*

```
>>> 'one' in eng2sp
True
>>> 'uno' in eng2sp
False
```

```
>>> vals = eng2sp.values()
>>> 'uno' in vals
True
```

Example: histogram

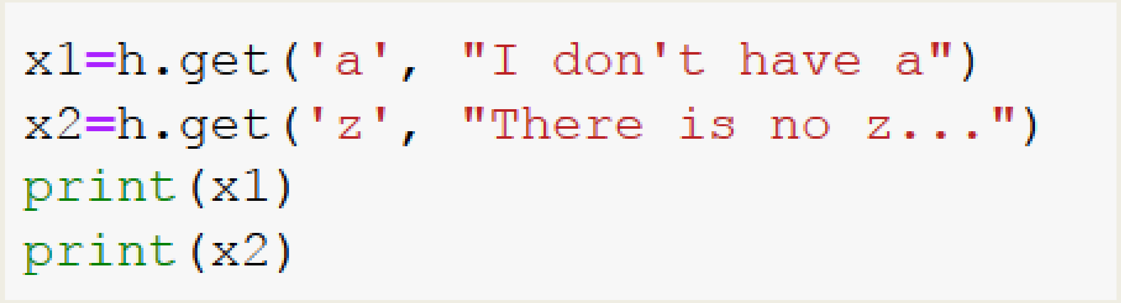
- Count the number of characters
  - *Dictionary*
    - Characters as keys
    - Counters as corresponding values

```
def histogram(s):
    d = dict()
    for c in s:
        if c not in d:
            d[c] = 1
        else:
            d[c] += 1
    return d
```

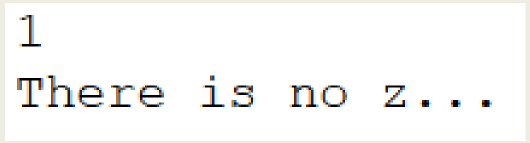
```
>>> h = histogram('brontosaurus')
>>> h
{'a': 1, 'b': 1, 'o': 2, 'n': 1, 's': 2, 'r': 2, 'u': 2, 't': 1}
```

Dictionary Method: get()

- .get()
  - *Input*
    - Key
    - Default value
  - *if key exists,*
    - return value
  - *else*
    - Default value



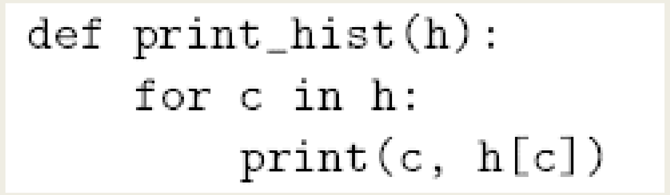
```
x1=h.get('a', "I don't have a")
x2=h.get('z', "There is no z...")
print(x1)
print(x2)
```



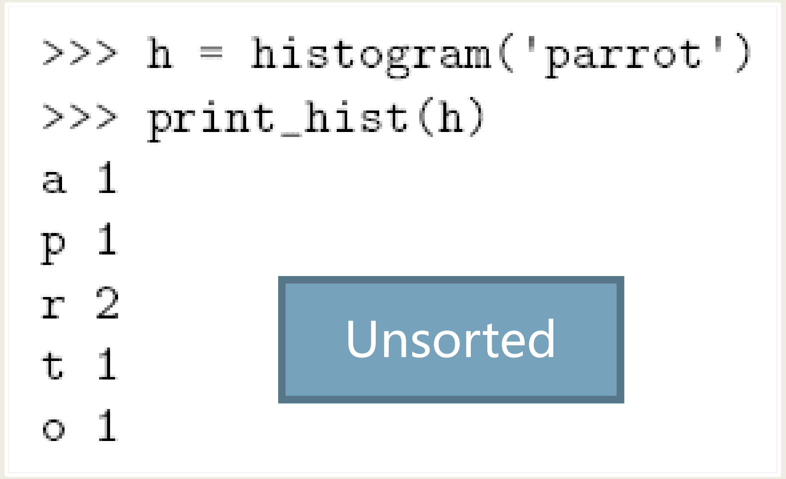
```
1
There is no z...
```

Unsorted v.s. Sorted results

- Key-value pairs do **NOT** have specific order in dictionary
- If you want to display sorted key-value pairs
  - *Do it yourself with repetition*
    - **sorted()**
    - **in**

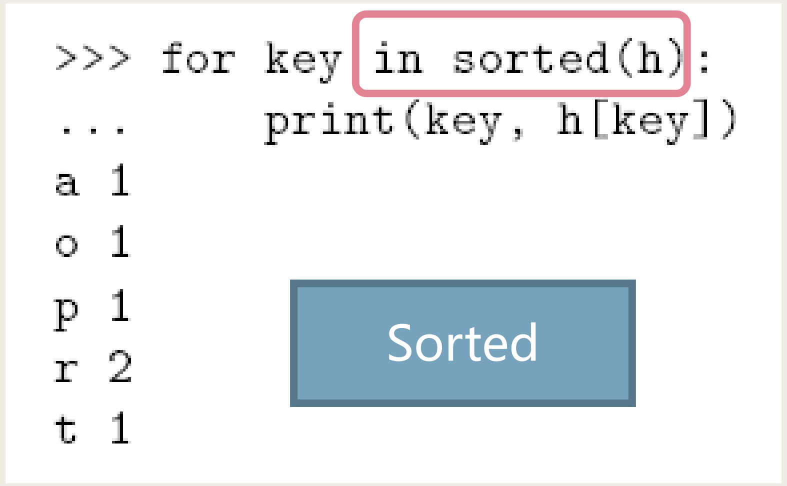


```
def print_hist(h):
    for c in h:
        print(c, h[c])
```



```
>>> h = histogram('parrot')
>>> print_hist(h)
a 1
p 1
r 2
t 1
o 1
```

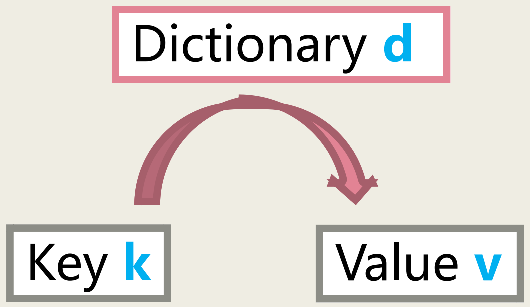
Unsorted



```
>>> for key in sorted(h):
...     print(key, h[key])
a 1
o 1
p 1
r 2
t 1
```

Sorted

Lookup and Reverse Lookup



- Lookup
  - *Use key to find value*

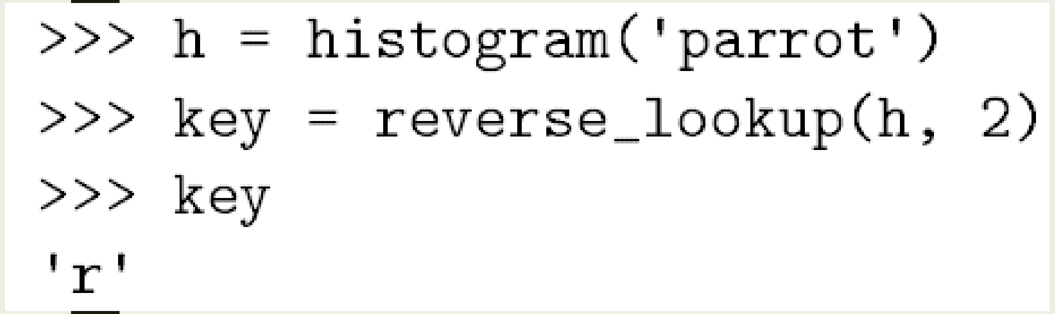
```
v = d[k]
```

- Reverse Lookup
  - *Use value to find key*

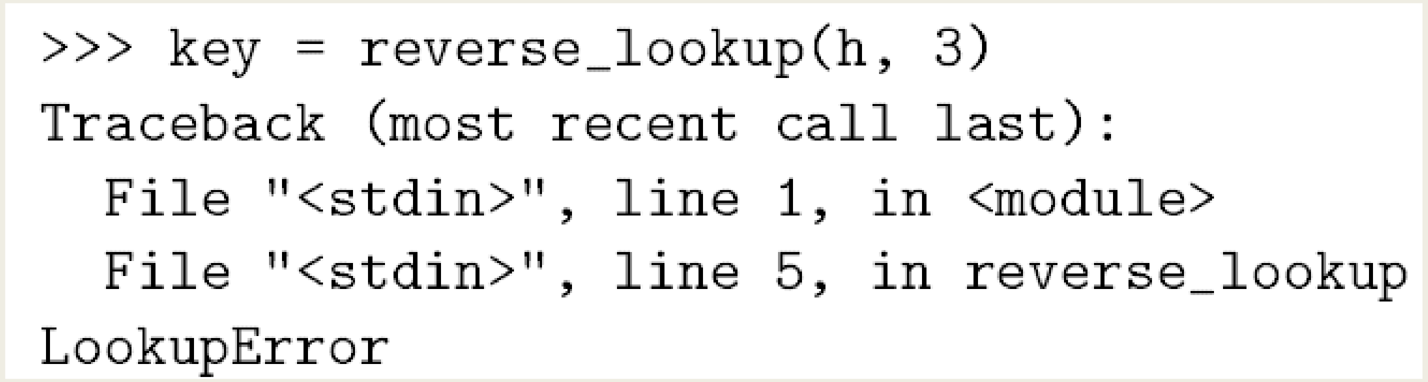
```
def reverse_lookup(d, v):
    for k in d:
        if d[k] == v:
            return k
    raise LookupError()
```

Exception handling: raise statement

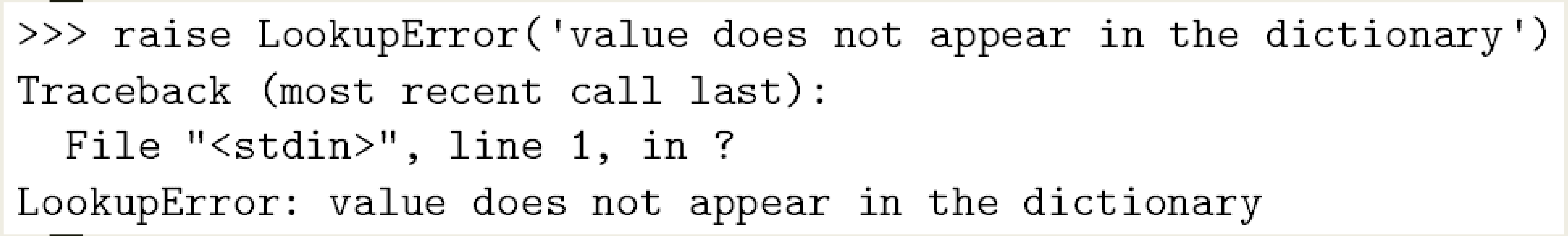
- Raise an exception
  - *Print traceback and an error message (like typical Python error)*
    - Customize your error message



```
>>> h = histogram('parrot')
>>> key = reverse_lookup(h, 2)
>>> key
'r'
```



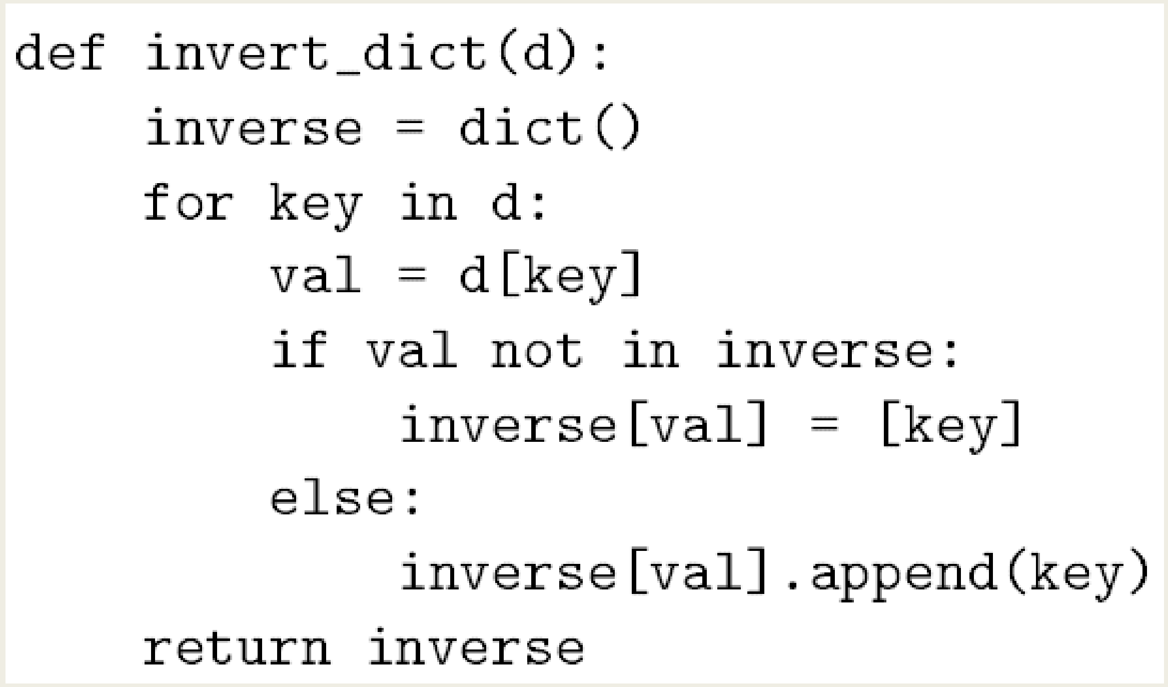
```
>>> key = reverse_lookup(h, 3)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "<stdin>", line 5, in reverse_lookup
LookupError
```



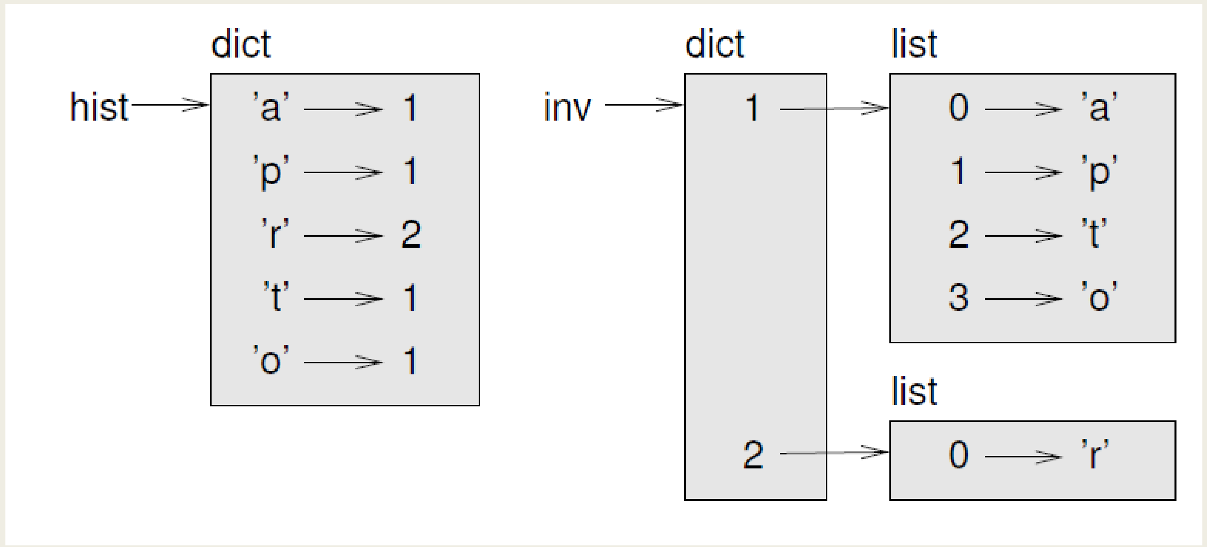
```
>>> raise LookupError('value does not appear in the dictionary')
Traceback (most recent call last):
  File "<stdin>", line 1, in ?
LookupError: value does not appear in the dictionary
```

Use **list** as a value in **dictionary**

- Example
  - *Dictionary maps letters to frequencies*
  - *Invert this dictionary*
    - Frequencies map to letters
      - *You might have the same frequency values for several letters*



```
def invert_dict(d):
    inverse = dict()
    for key in d:
        val = d[key]
        if val not in inverse:
            inverse[val] = [key]
        else:
            inverse[val].append(key)
    return inverse
```



```
>>> hist = histogram('parrot')
>>> hist
{'a': 1, 'p': 1, 'r': 2, 't': 1, 'o': 1}
>>> inverse = invert_dict(hist)
>>> inverse
{1: ['a', 'p', 't', 'o'], 2: ['r']}
```

More about list and dictionary

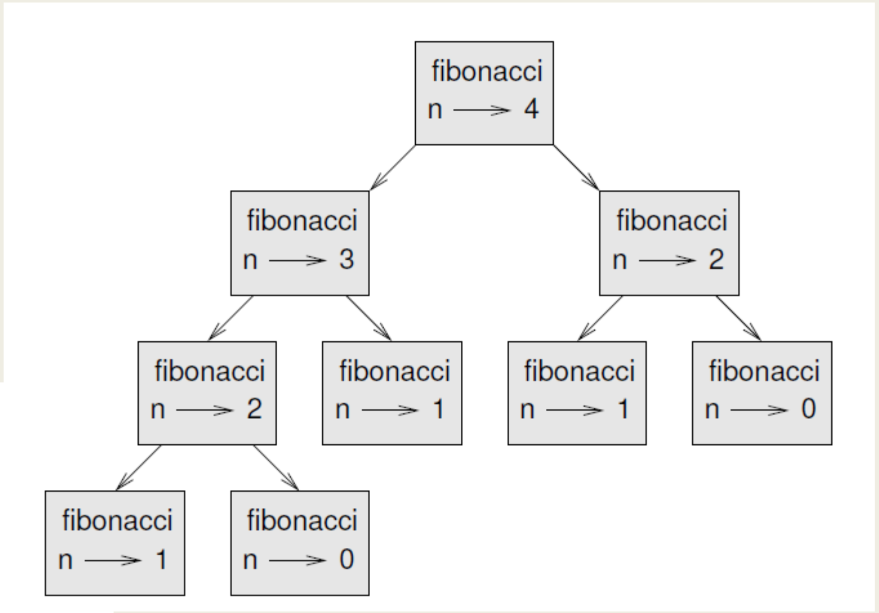
- Lists can be values in a dictionary
  - *As shown in previous example*
- Lists **cannot** be keys

- Dictionary is implemented using a hashtable
  - *Hash function that gets a value and returns an integer*
  - *Keys must be hashable*
    - Do not use mutable type (e.g. list) for keys

Memos

- Memoized version of fibonacci

```
known = {0:0, 1:1}

def fibonacci(n):
    if n in known:
        return known[n]

    res = fibonacci(n-1) + fibonacci(n-2)
    known[n] = res
    return res
```



Recall: Section 6.7 Recursion

Global variables

- Global variables v.s. local variables in a function
- Use global variables carefully
  - *Examples*
    - Flag
      - *Verbose*
      - *Tracking whether a function has been called*

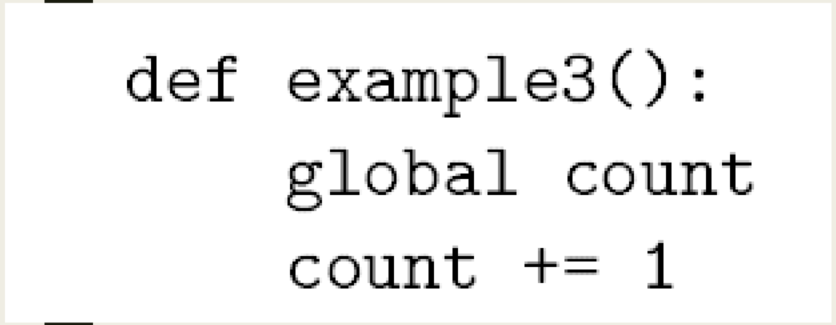
```
verbose = True

def example1():
    if verbose:
        print('Running example1')
```

- Use global variable within a function
  - *Declare the global variable*

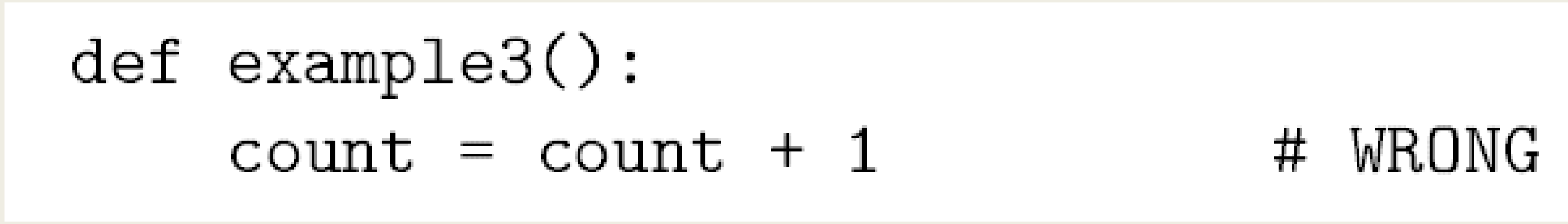
```
been_called = False

def example2():
    global been_called
    been_called = True
```

```
def example2():
    been_called = True          # WRONG
```

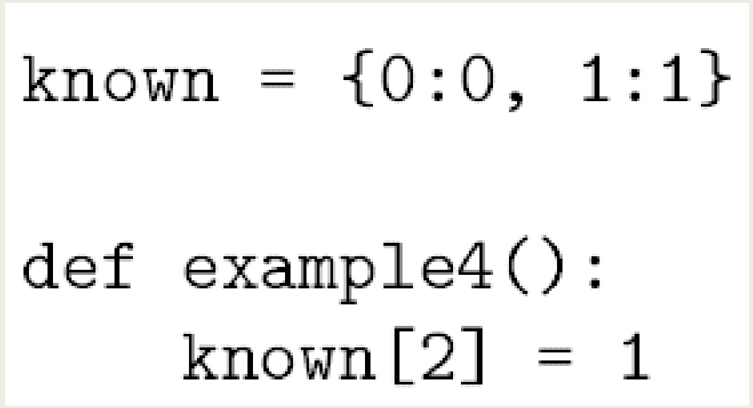
More global variables



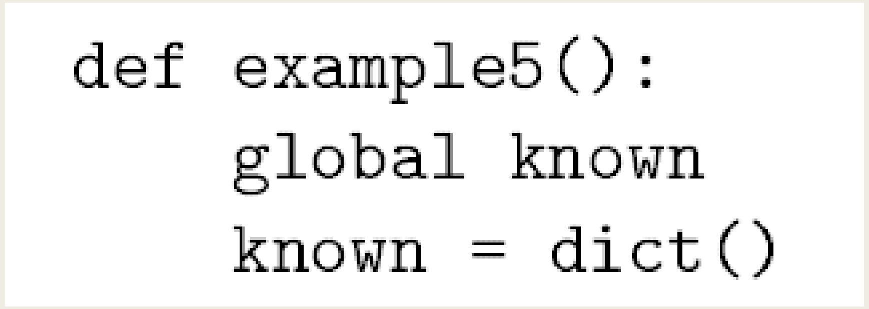
```
def example3():
    global count
    count += 1
```



```
def example3():
    count = count + 1            # WRONG
```



```
known = {0:0, 1:1}

def example4():
    known[2] = 1
```



```
def example5():
    global known
    known = dict()
```

Tips for debugging

- Scale down the input
- Check summaries and types
- Write self-checks
- Format the output

Reading

- Chapter 11 in textbook “Think Python”