Scripting Languages

Python basics

- dictionaries
- iteration over dictionary elements
- exercise

Data type → `dictionary`

Unordered collection of items (*associative memory*)

- a set of pairs: `key: value`
  - keys must be **unique** (numbers, strings, tuples, **but not lists**)
  - values can be anything (including lists and dictionaries)
- can be modified (grow, shrink , item changes)
- can be joined, compared, **but not sliced**

```
x = {key1: value1, key2: value2, …}
```

```
>>> lookup = {'one':1, 'two':2, 'three':3}
>>> shopping = {'orange': 1, 'milk': '2 cans'}
>>> x = {"tom":"tom@car.toon","jerry":"jerry@car.toon"}

>>> print len(lookup), len(shopping), len(x)
3 2 2
```

dictionary → creating

- empty dictionary

```
d = {}  |  d = dict()
```

- explicitly

```
d = { key : value, … }
```

- by mapping

```
d = dict( [ (key, value), … ] }
d = dict( zip( [ key, … ], [ value, … ] ) )
```

```
d1 = {'one': 1, 'two': 2, 'three': 3}

d2 = dict([('one', 1), ('two', 2), ('three', 3)])

d3 = dict(zip(['one', 'two', 'three'], [1, 2, 3]))
```

dictionary → data access

- items can be read/modified by key

```
x = d[key]  or  d[key]=x
```

- items (or dictionary) can be added (or merged)

```
d[newkey] = x
d.update(newdict)
```

- items can be deleted (or removed all)

```
del d[key]
d.clear()
```

```
>>> lookup = {'three':3, 'two':2, 'one':1}

>>> lookup['four']=4
>>> del lookup['one']

>>> print(lookup, len(lookup))
{'four': 4, 'three': 3, 'two': 2} 3
```

dictionary → as database

- checking membership of a key

```
key in d        → True/False
key not in d    → False/True
```

- add new or update existing items

```
if key not in d:        |       if not (key in d):
    d[key] = initial value
else:
    modify existing d[key]
```

```
weapons = {'knife':2, 'sword':1, 'gun':3}
found = 'axe'

if found not in weapons:
    weapons[found] = 1
else:
    weapons[found] += 1
```

`dictionary` → view objects

- iterable objects for dictionaries (always up-to-date):
  - `d.keys()` → view of keys (all elements unique)
  - `d.values()` → view of values (possible duplicates)
  - `d.items()` → view of key:value pairs
  - views support length and membership test
  - views can be converted to lists → `list(view)`

```
>>> students = {'Alice': 1, 'James': 2, 'John': 1}
>>> names = students.keys()

>>> print(list(names), len(names))
['Alice', 'James', 'John'] 3

>>> del students["Alice"]
>>> print(list(names), len(names))
['James', 'John'] 2
```

`dict.` → iterating over keys

- applying operation to all dictionary elements
  - !!! keys are iterated in unknown order:

```
for key in d.keys():
    do something with d[key]
```

```
students = {'Alice': 1, 'James': 2, 'John': 1}

for name in students.keys():
    if students[name] == 1:
        print("Unique name: ", name)
```

```
Unique name: Alice
Unique name: John
```

`dict.` → iterating over values

- examining all dictionary values (possible duplicates)

```
for val in d.values():
    do something with val
```

```
min|max|sum(d.values())
```

```
students = {'Alice': 1, 'James': 2, 'John': 1}

all_students = sum(students.values())
most_common = max(students.values())

for name in students.keys():
    if students[name] == most_common:
        print(name, "is the most common name")
        print(most_common, " times out of ", all_students)
```

```
James is the most common name
2 times out of 4
```

`dict.` → iterating over items

- easiest way to access whole dictionary content
  - iteration over **key:value** pairs (in unknown order)

```
for key,val in d.items():
    do something with key,val
```

```
students = {'Alice':1, "James":2, 'John': 1}
occurrences = {}

for name, occ in students.items():
    if occ not in occurrences:
        occurrences[occ]=[name]
    else:
        occurrences[occ].append(name)

print(occurrences)
```

reversing the dictionary

```
{1: ['John', 'Alice'], 2: ['James']}
```

Exercise → text statistics

Perform the letters-and-words content analysis of text

- text file "dorian.txt"
  - "The picture of Dorian Gray" by Oscar Wilde, 1890
  - english, 8600 lines, pure ASCII content
  - analysis: case-insensitive, with all punctuation characters striped

Questions:

- what are the most/least common letters in the text
- how many unique words are in the text
- which word is the most common (top-10 most common words)
- which word is the longest (top-10 longest words)

Exercise → letters stats

1) count occurrences for all letters in the text
   - `locc` → storing occurrences of letters
     - example: `locc = {'e': 3458, 'h': 167, …}`

```
locc = {} # dictionary for occurrences of letters

file = open("dorian.txt", 'r')

for line in file:
    letters = line.strip().lower()
    for l in letters:
        if l not in locc:
            locc[l] = 1
        else:
            locc[l] += 1

file.close()
```

Exercise → letters stats

2) reverse the dictionary

- `rlocc` → lists of letters with the same occurrence
  - example:
    `rlocc = {3458:['e'], 167:['h', 's'], …}`

```
# Reversing locc-dictionary

rlocc = {}

for l, f in locc.items():
    if f not in rlocc:
        rlocc[f] = [l]
    else:
        rlocc[f].append(l)
```

Exercise → letters stats

3) print the results from: `rlocc`

- sorted by occurrence (keys)
- in reversed order (for the most common first)

ver. 1 – making new lists

```
for f in reversed(sorted(rlocc.keys())):
    print(rlocc[f], f)
```

ver. 2 – transforming the list in place

```
x = list(rlocc.keys())
x.sort()
x.reverse()
for f in x:
    print(rlocc[f], f)
```

Exercise → letters stats

```
locc = {}

file = open("dorian.txt", 'r')
for line in file:
    letters = line.strip().lower()
    for l in letters:
        if l not in locc:
            locc[l] = 1
        else:
            locc[l] += 1
file.close()

# Reversing locc-dictionary
rlocc = {}
for l, f in locc.items():
    if f not in rlocc:
        rlocc[f] = [l]
    else:
        rlocc[f].append(l)

# Printing letter-statistics results
for f in reversed(sorted(rlocc.keys())):
    print(rlocc[f], f)
```

Exercise → words stats

- create dictionaries:
  - wocc → storing the occurrences of (unique) words
    - example: `wocc = {'the': 3458, 'hello': 167, …}`
  - wsize →storing the lengths of unique words
    - example: `wsize = {'the': 3, 'hello': 5, …}`
- create reverse dictionaries
  - rwocc → storing the list of words with the same occurrences
    - example: `rwocc = {3458:['the', 'an',], 167:['hello', 'sir'], …}`
  - rwsize →storing the list of words having the same length
    - example: `rwsize = {3: ['the', 'and'], 5: ['hello', 'about'], …}`
- generate lists `rwocc.keys()` and `rwsize.keys()`
  - sort the keys-lists, reverse the order, print top-20 of this list

Exercise → words stats

- read text line-by-line → **`for line in file:`**
- clean line and convert to low-caps → **`strip()`, `lower()`**
- break a text line into a list of words → **`split()`**
- strip unwanted characters and punctuation → **`strip(chars)`**

```
wocc = {}
wsize = {}

file = open("dorian.txt", 'r')
for line in file:
    words = line.strip().lower().split()
    for w in words:
        w = w.strip(".,;:-'`\"?!()")
        if not (w in wocc):
            wocc[w] = 1
            wsize[w] = len(w)
        else:
            wocc[w] += 1
file.close()
```

2) reverse the dictionaries, 3) print the results

Exercise → results

letters statistics:

```
[' '] 71663   ...
['e'] 40243   ['q'] 323
['t'] 29013   ['j'] 301
['a'] 27309   [';'] 167
['o'] 25990   ['z'] 163
['i'] 23333   [':'] 51
['h'] 21825   ['1'] 4
['n'] 21335   ['2'] 3
['s'] 20372   ['8', '5', '0'] 2
['r'] 19376   ['9'] 1
...
```

words statistics:

unique words = 7133

3731 ['the']	994 ['his']
2196 ['and']	886 ['is']
2152 ['of']	832 ['had']
2035 ['to']	661 ['him']
1686 ['i']	659 ['with']
1655 ['a']	578 ['for']
1538 ['he']	567 ['as']
1442 ['you']	562 ['have']
1354 ['that']	560 ['at']
1338 ['it']	521 ['me']
1200 ['in']	470 ['not']
1084 ['was']	...

23 ['elaborately-illustrated']
22 ['opal-and-iris-throated']
21 ['fantastically-painted', 'luxuriously-cushioned']
19 ['seventeenth-century']
18 ['vermilion-and-gold', 'pistachio-coloured', 'delicately-scented', 'nineteenth-century']
17 ['cunningly-wrought', 'well-proportioned', 'brightly-coloured']
16 [..., 'responsibilities', ...]
15 [..., 'extraordinarily', ...]