INFO-H-600 - Computing foundations of data sciences

Session 4
Introduction to Python
Sets, dictionaries and files

Université libre de Bruxelles
École polytechnique de Bruxelles

2019-2020

Sets

Strings, tuples and lists are sequences

- they are ordered : $l[i]$ acces the ith element of l

Sets are unordered datastructures !

- mutable : elements can be added and removed after creation
- no order
- no duplicates

Sets : examples from python.org

```
>>> basket = {'apple', 'orange', 'apple', 'pear',
                                              'orange', 'banana'}
>>> print(basket)                  # duplicates have been removed
{'orange', 'banana', 'pear', 'apple'}
>>> 'orange' in basket             # fast membership testing
True

>>> a = set('abracadabra')
>>> b = set('alacazam')
>>> a                                  # unique letters in a
{'a', 'r', 'b', 'c', 'd'}
>>> a - b                              # letters in a but not in b
{'r', 'd', 'b'}
>>> a | b                              # letters in a or b or both
{'a', 'c', 'r', 'd', 'b', 'm', 'z', 'l'}
>>> a & b                              # letters in both a and b
{'a', 'c'}
>>> a ^ b                              # letters in a or b but not both
{'r', 'd', 'b', 'm', 'z', 'l'}
```

Sets : iteration

You can iterate on sets . . .

```
for fruit in basket:
    print(fruit, end=" ")
```

. . . but no order is not guarenteed !

- could print : orange banana pear apple
- could print : pear banana apple orange
- could print : orange apple pear banana
- . . .

Dictionary

A dictionairy is a type of mutable datasets.

Unlique sequences (lists, strings, tupples) which are indexed using theire position in the datastructure, dictionaries are unordered and indexed using keys.

Dictionairies are an unordered set of paires *key-value*.

```
>>> tel = {}
>>> tel["Gary"] = 3766
>>> tel["Thierry"] = 5603
>>> tel["Alain"] = 1234
>>> print(tel)
{'Alain': 1234, 'Thierry': 5603, 'Gary': 3766}
```

Keys must be immutables (numbers, string, tuples). Values can be anything.

Operations on dictionaries

```
>>> tel.items()                          # Iterator on the elements
dict_items([('Alain', 1234), ('Thierry', 5603), ('Gary', 3766)])
>>> list(tel.items())                    # List of elements
[('Alain', 1234), ('Thierry', 5603), ('Gary', 3766)]
>>> tel.keys()                           # Iterator on the keys
dict_keys(['Alain', 'Thierry', 'Gary'])
>>> list(tel.keys())                     # List of keys
['Alain', 'Thierry', 'Gary']
>>> tel.values()                         # Iterator on the vlaues
dict_values([1234, 5603, 3766])
>>> list(tel.values())                   # List of values
[1234, 5603, 3766]
>>> for cle in tel:                      # Iteration on the keys
...         print(cle + ": " + str(tel[cle]))
...
Alain: 1234
Thierry: 5603
Gary: 3766
```

Operations on dictionaries

```
>>> "Gary" in tel                          # key exists test
True
>>> del tel["Thierry"]                     # deletion (similar as sequences
>>> print(tel)
{'Alain': 1234, 'Gary': 3766}
>>> print(tel["Thierry"])                  # Unexisting key -> Error

...KeyError: 'Thierry'
>>> print(tel.get("Thierry"))              # Unexisting key -> None
None
>>> print(tel.get("Thierry", 1307))        # Unexisting key -> ...
1307                                       # value choosen
```

Example

```
def occurrences(mot):
    d = {}
    for c in mot:
        d[c] = d.get(c, 0) + 1
    return d

print(occurrences("banane"))
```

shows

```
{'a': 2, 'b': 1, 'e': 1, 'n': 2}
```

Reading Files

Suppose the `info.txt` file located in the current directiory containing the following lines :

```
Anh Vu
Gary
```

```
>>> f = open("info.txt")
>>> f.read()
'Anh Vu\nGary'
```

```
>>> f = open("info.txt")
>>> f.readlines()
['Anh Vu\n', 'Gary']
```

```
>>> f = open("info.txt")
>>> f.readline()
'Anh Vu\n'
>>> f.readline()
'Gary'
>>> f.readline()
''
```

```
>>> f = open("info.txt")
>>> for l in f.readlines():
...     l
...
'Anh Vu\n'
'Gary'
```

Notice the presence of the end of line indicator in some results. `strip()` allows to get rid of it.

```
>>> f = open("info.txt")
>>> f.readline().strip()
'Anh Vu'
```

Writing files

Append mode (add at the end of file)

```
>>> f = open("info.txt","a")
>>> f.write("\nGary")
>>> f.close()                    # Finalise writing
```

File content :

```
Anh Vu
Gary
Gary
```

Write mode (replace content) :

```
>>> f = open("info.txt","w")
>>> f.write("Alain")
>>> f.close()
```

File content :

```
Alain
```

Reminder : some functions on strings

```
>>> s = "  \n  Foo\nBar spam\n"
>>> s
'  \n  Foo\nBar spam\n'
>>> s.upper()
'  \n  FOO\nBAR SPAM\n'
>>> s.lower()
'  \n  foo\nbar spam\n'
>>> s.strip()                # Clean start and end
'Foo\nBar spam'
>>> s.replace("\n", "-") # Remplacement
'  -  Foo-Bar spam-'
>>> s.replace("\n", "")  # Deletes
'   FooBar spam'
>>> s.split()                # Removes whitespaces
['Foo', 'Bar', 'spam']
>>> s.split("\n")
['  ', '  Foo', 'Bar spam', '']
```

Careful, *strings* are immutables. These funcions return copies.