UNIVERSITÉ DE LORRAINE

Institut des sciences du Digital Management & Cognition
COMPOSANTE DE L'UNIVERSITÉ DE LORRAINE

Programming

3- lists again, dictionaries

Those slides will be available on Arche

More on Lists

TODAY

Lists again

Dictionaries

Lists

Lists in python are order **sets/sequences of values**. They are used to store multiple items in a single variable.

```
[1,2,2,4]
list(("pierre-baptiste", "amélie", "belen", "mehsen", "pin-xun", "camille"))
[] # the empty list
[1, "bob", -3.5, 2+3j, True]
```

Lists are **ordered**, **changeable** and **removable**.

Accessing elements of a list

```
mylist = ["pierre-baptiste", "amélie", "belen", "mehsen", "pin-xun", "camille"]
type(mylist) #>>> mylist
len(mylist)  #>>> 6
mylist[0]    #>>> “pierre-baptiste”
mylist[1]    #>>> “amélie”
mylist[3]    #>>> “mehsen”
mylist[-1]   #>>> “camille”
mylist[-3]   #>>> “mehsen”
```

Substrings and slicing

Given a list, such as

```
mylist = ["pierre-baptiste", "amélie", "belen", "mehsen", "pin-xun", "camille"]
```

`mylist[start:stop:inc]` provides a way to obtain a substring.

`start` - index of the start of the substring (0 by default)

`stop` – index before which to stop (`len(list)` by default)

`inc` – increment of the index (1 by default).

Substrings and slicing

Given a list, such as

```
mylist = ["pierre-baptiste", "amélie", "belen", "mehsen", "pin-xun", "camille"]
```

`mylist[start:stop:inc]` provides a way to obtain a substring.

```
mylist[:]
#>>> ["pierre-baptiste", "amélie", "belen", "mehsen", "pin-xun", "camille"]
```

Substrings and slicing

Given a list, such as

```
mylist = ["pierre-baptiste", "amélie", "belen", "mehsen", "pin-xun", "camille"]
```

`mylist[start:stop:inc]` provides a way to obtain a substring.

`mylist[2:]`

Substrings and slicing

Given a list, such as

```
mylist = ["pierre-baptiste", "amélie", "belen", "mehsen", "pin-xun", "camille"]
```

`mylist[start:stop:inc]` provides a way to obtain a substring.

```
mylist[2:]

#>>> ["belen", "mehsen", "pin-xun", "camille"]
```

Substrings and slicing

Given a list, such as

```
mylist = ["pierre-baptiste", "amélie", "belen", "mehsen", "pin-xun", "camille"]
```

`mylist[start:stop:inc]` provides a way to obtain a substring.

`mylist[2:4]`

Substrings and slicing

Given a list, such as

```
mylist = ["pierre-baptiste", "amélie", "belen", "mehsen", "pin-xun", "camille"]
```

`mylist[start:stop:inc]` provides a way to obtain a substring.

```
mylist[2:4]

#>>> ["belen", "mehsen"]
```

Substrings and slicing

Given a list, such as

```
mylist = ["pierre-baptiste", "amélie", "belen", "mehsen", "pin-xun", "camille"]
```

`mylist[start:stop:inc]` provides a way to obtain a substring.

`mylist[:4]`

Substrings and slicing

Given a list, such as

```
mylist = ["pierre-baptiste", "amélie", "belen", "mehsen", "pin-xun", "camille"]
```

`mylist[start:stop:inc]` provides a way to obtain a substring.

```
mylist[:4]

#>>> ["pierre-baptiste", "amélie", "belen", "mehsen"]
```

Substrings and slicing

Given a list, such as

```
mylist = ["pierre-baptiste", "amélie", "belen", "mehsen", "pin-xun", "camille"]
```

`mylist[start:stop:inc]` provides a way to obtain a substring.

`mylist[1:4:2]`

Substrings and slicing

Given a list, such as

```
mylist = ["pierre-baptiste", "amélie", "belen", "mehsen", "pin-xun", "camille"]
```

`mylist[start:stop:inc]` provides a way to obtain a substring.

```
mylist[1:4:2]

#>>> ["amélie", "mehsen"]
```

Substrings and slicing

Given a list, such as

```
mylist = ["pierre-baptiste", "amélie", "belen", "mehsen", "pin-xun", "camille"]
```

`mylist[start:stop:inc]` provides a way to obtain a substring.

`mylist[1::2]`

Substrings and slicing

Given a list, such as

```
mylist = ["pierre-baptiste", "amélie", "belen", "mehsen", "pin-xun", "camille"]
```

`mylist[start:stop:inc]` provides a way to obtain a substring.

```
mylist[1::2]

#>>> ["amélie", "mehsen", "camille"]
```

Substrings and slicing

Given a list, such as

```
mylist = ["pierre-baptiste", "amélie", "belen", "mehsen", "pin-xun", "camille"]
```

`mylist[start:stop:inc]` provides a way to obtain a substring.

`mylist[::-1]`

Substrings and slicing

Given a list, such as

```
mylist = ["pierre-baptiste", "amélie", "belen", "mehsen", "pin-xun", "camille"]
```

`mylist[start:stop:inc]` provides a way to obtain a substring.

```
mylist[::-1]
#>>> [“camille”, “pin-xun”, “mehsen”, “belen”, “amélie”, “pierre-baptiste”]
```

Substrings and slicing

Given a list, such as

```
mylist = ["pierre-baptiste", "amélie", "belen", "mehsen", "pin-xun", "camille"]
```

`mylist[start:stop:inc]` provides a way to obtain a substring.

`mylist[::-2]`

Substrings and slicing

Given a list, such as

```
mylist = ["pierre-baptiste", "amélie", "belen", "mehsen", "pin-xun", "camille"]
```

`mylist[start:stop:inc]` provides a way to obtain a substring.

```
mylist[::-2]

#>>> ['camille', 'mehsen', 'amélie']
```

Substrings and slicing

Given a list, such as

```
mylist = ["pierre-baptiste", "amélie", "belen", "mehsen", "pin-xun", "camille"]
```

`mylist[start:stop:inc]` provides a way to obtain a substring.

```
x = 1

mylist[x:25-21]
```

Substrings and slicing

Given a list, such as

```
mylist = ["pierre-baptiste", "amélie", "belen", "mehsen", "pin-xun", "camille"]
```

`mylist[start:stop:inc]` provides a way to obtain a substring.

```
x = 1

mylist[x:25-21]

#>>> ['amélie', 'belen', 'mehsen']
```

Modifying lists

```
mylist[2] = "kira"
#>>> ['pierre-baptiste', 'amélie', 'kira', 'mehsen', 'pin-xun', 'camille']
mylist[2:5] = ["aurore", "solène", "nassim"]
#>>> ['pierre-baptiste', 'amélie', 'aurore', 'solène', 'nassim', 'camille']
mylist.append("pauline")
```

(equivalent to list[len(list):] = [“pauline”])

```
#>>> ['pierre-baptiste', 'amélie', 'aurore', 'solène', 'nassim', 'camille', 'pauline']
mylist.extend(["alberto", "telma"])
```

(equivalent to list[len(list):] = [“alberto”, “telma”])

```
#>>> ['pierre-baptiste', 'amélie', 'aurore', 'solène', 'nassim', 'camille', 'pauline', 'alberto',
'telma']
```

By the way

```
mylist = ["pierre-baptiste", "amélie", "belen", "mehsen", "pin-xun", "camille"]
```

What will happen if I do:

```
mylist.extend("guilherme")
```

Which is equivalent to:

```
mylist[len(mylist):] = "guilherme"
```

By the way

```
mylist = ["pierre-baptiste", "amélie", "belen", "mehsen", "pin-xun", "camille"]
```

What will happen if I do:

```
mylist.extend("guilherme")
```

```
#>>> ['pierre-baptiste', 'amélie', 'belen', 'mehsen', 'pin-xun', 'camille', 'g', 'u', 'i', 'l', 'h', 'e', 'r', 'm', 'e']
```

Which is equivalent to:

```
mylist[len(mylist):] = "guilherme"
```

```
#>>> ['pierre-baptiste', 'amélie', 'belen', 'mehsen', 'pin-xun', 'camille', 'g', 'u', 'i', 'l', 'h', 'e', 'r', 'm', 'e']
```

Modifying lists

```
mylist = ["pierre-baptiste", "amélie", "belen", "mehsen", "pin-xun", "camille"]
mylist.insert(3, "paul")
#>>> ['pierre-baptiste', 'amélie', 'belen', 'paul', 'mehsen', 'pin-xun', 'camille']
del mylist[2]
#>>> ['pierre-baptiste', 'amélie', 'paul', 'mehsen', 'pin-xun', 'camille']
del mylist[2:5]
#>>> ['pierre-baptiste', 'amélie', 'camille']
mylist.remove("amélie")
#>>> ['pierre-baptiste', 'camille']
```

Other common operations on lists

```
mylist = ['pierre-baptiste', 'amélie', 'aurore', 'solène', 'nassim', 'camille', 'pauline', 'alberto', 'telma']

"amélie" in mylist  #>>> True

"amelie" in mylist  #>>> False

mylist.reverse() #>>> ['telma', 'alberto', 'pauline', 'camille', 'nassim', 'solène', 'aurore', 'amélie', 'pierre-baptiste']

x = mylist.pop()

#>>> ['telma', 'alberto', 'pauline', 'camille', 'nassim', 'solène', 'aurore', 'amélie']

#>>> 'pierre-baptiste'

x = mylist.index("pauline") #>>> 2

mylist.count("camille") #>>> 1

mylist = ["amélie", “camille”, "pierre-baptiste", "belen", "mehsen", "pin-xun", "camille"]

mylist.count("camille") #>>> 2
```

By the way…

How can you write 2 instructions that do the same as

```
x = list.pop()
```

using list access and slicing?

Sorting

```
mylist = ['aurore', 'solène', 'nassim', 'camille', 'pauline', 'alberto', 'telma']
mylist.sort()
#>>> ['alberto', 'aurore', 'camille', 'nassim', 'pauline', 'solène', 'telma']
mylist.sort(reverse=True)
#>>> ['telma', 'solène', 'pauline', 'nassim', 'camille', 'aurore', 'alberto']
```

Importance of understanding pointers for lists

```
list1 = [1,2,3]
list2 = list1
list3 = [1,2,3]
list1 == list3 >>> True
list1 is list2 >>> True
list1 is list3 >>> False

list1[1] = 4
list1 >>> [1,4,3]
list2 >>> [1,4,3]
```

Dictionaries

Dictionaries (dict)

A dictionary is a structured object where information is organised according to keys. For example in

`{“a”: value1, “b”:, value2, “c”: value3}`

`a`, `b` and `c` are keys (attributes) and `value1`, `value2` and `value3` are the values associated with those keys.

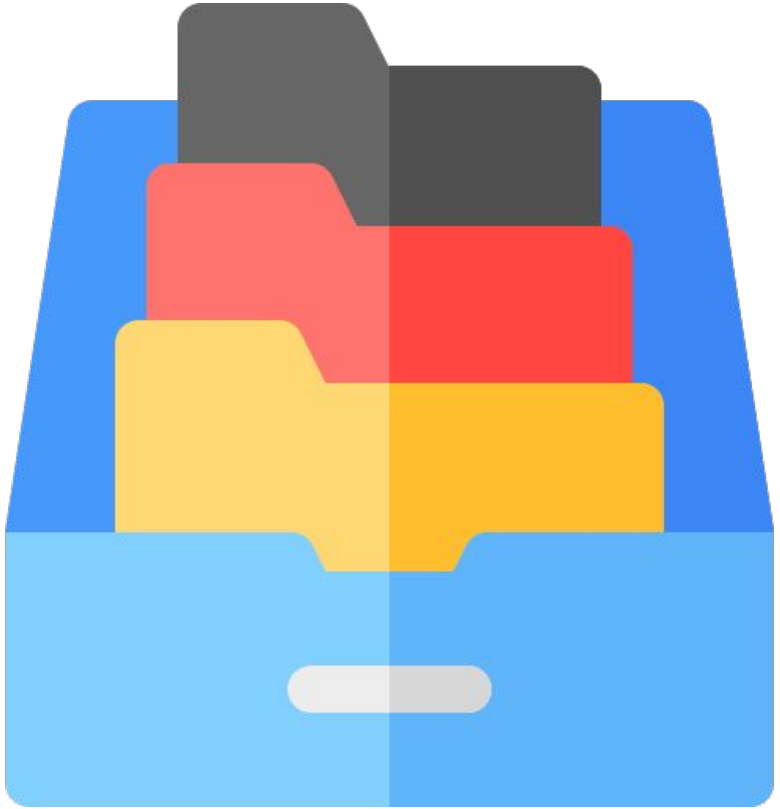
It can be seen as equivalent (more or less) to an associative array in other languages.

Dictionaries (dict)

- Dictionaries are another Type of variable `dict()`
- Follow only one concept: **Key and Value**
- Can contain deeper informations, with explicit hierarchy
- Data are not ordered (contrary to `list()`)
- Value can be accessed by its associated Key
- **Keys are unique** you cannot associate multiple values to one key.
- `keys()` returns the list of keys, `values()` returns the list of values (you just need to change its type to `list()`)

Dictionaries (dict)

How to retrieve the yellow file?

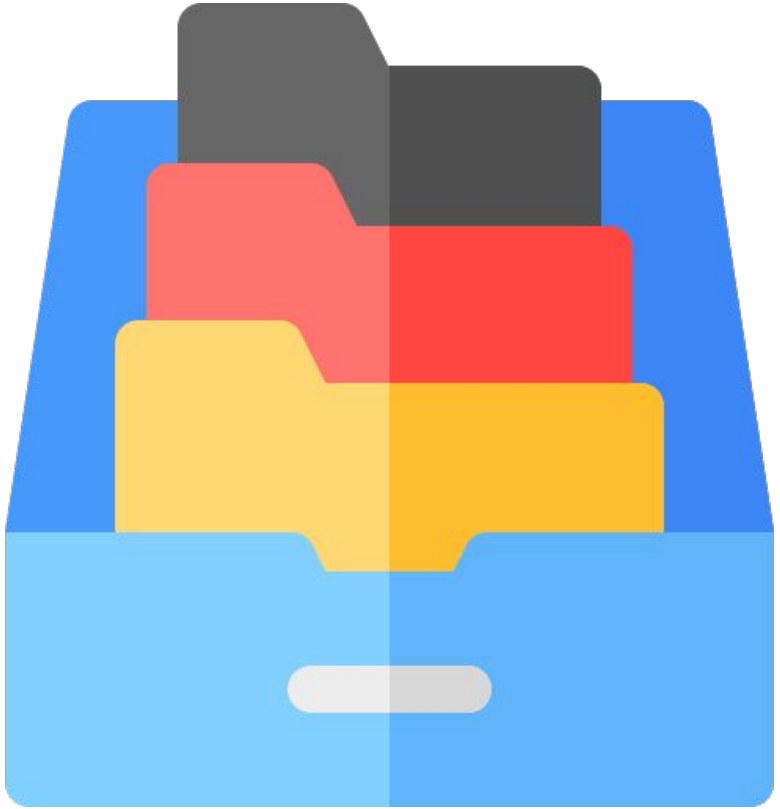


Dictionaries (dict)

How to retrieve the yellow file?

Keys = **“red”**, **“yellow”**, **“black”**
Values = file content

```
drawer = {“red”:”content”,
“yellow”:”content”, “black”:”content”}
```

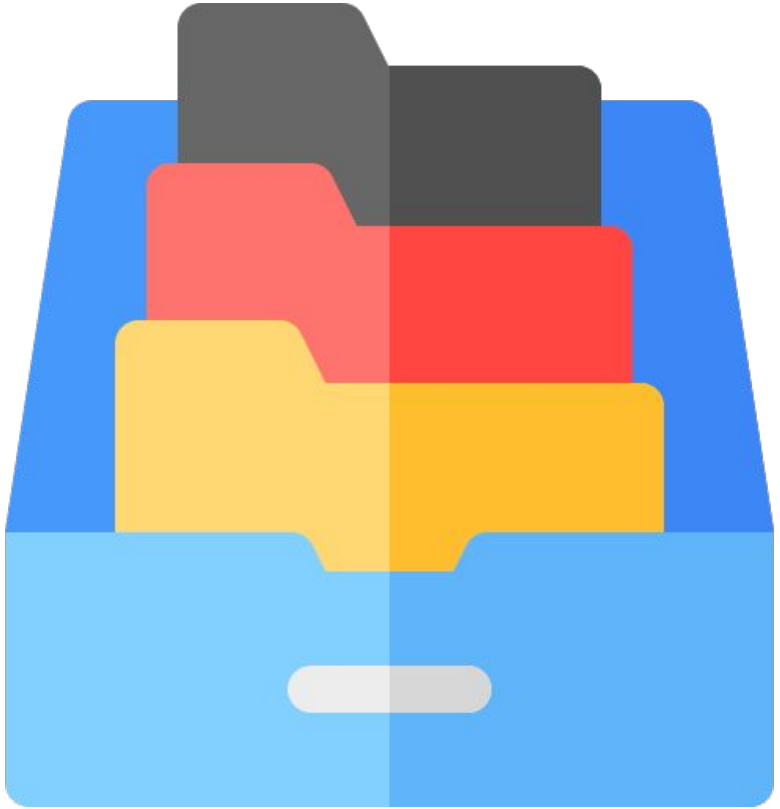


Dictionaries (dict)

How to retrieve the yellow file?

Keys = **“red”**, **“yellow”**, **“black”**
Values = file content

```
drawer = {“red”:”content”,
“yellow”:”content”, “black”:”content”}
print(drawer[“yellow”])
```



Dictionaries (dict)

{} are used to identify dict

A dict is just **a key associated with a value**

```
1  # initialize an empty dictionary
2  dico = {}
3  dico = dict() # both ways are corrects
4
5  # intialize with values
6  dico = { "key" : "value" }
7  print(dico)
```

Dictionaries (dict)

```
1 # more concrete example
2 player = { "name":"Alberto", "health":200 }
3 print(player)
4
5 student = {'name':'Camille', 'age':19, 'friends':['Pauline', 'Solène'] }
6 print(student)
```

Some basic operations on dictionaries

```
teacher = {
   "firstname": "John", "lastname": "Wick", "age": 52,
   "address": {
       "number": 34,"street": "rue Saint Bernard",
       "postcode": 57910,"town": "Berkange"},
   "courses": ["programming", "ai"]
}
type(teacher)
len(teacher)
len(teacher["address"])
"age" in teacher
"rage" in teacher
list(teacher)
```

Some basic operations on dictionaries

```
teacher = {
   "firstname": "John", "lastname": "Wick", "age": 52,
   "address": {
       "number": 34,"street": "rue Saint Bernard",
       "postcode": 57910,"town": "Berkange"},
   "courses": ["programming", "ai"]
}
type(teacher) #>>> dict
len(teacher) #>>> 5
len(teacher["address"])
"age" in teacher
"rage" in teacher
list(teacher)
```

Some basic operations on dictionaries

```
teacher = {
   "firstname": "John", "lastname": "Wick", "age": 52,
   "address": {
       "number": 34,"street": "rue Saint Bernard",
       "postcode": 57910,"town": "Berkange"},
   "courses": ["programming", "ai"]
}
type(teacher) #>>> dict
len(teacher) #>>> 5
len(teacher["address"]) #>>> 4
"age" in teacher
"rage" in teacher
list(teacher)
```

Some basic operations on dictionaries

```
teacher = {
   "firstname": "John", "lastname": "Wick", "age": 52,
   "address": {
       "number": 34,"street": "rue Saint Bernard",
       "postcode": 57910,"town": "Berkange"},
   "courses": ["programming", "ai"]
}
type(teacher) #>>> dict
len(teacher) #>>> 5
len(teacher["address"]) #>>> 4
"age" in teacher #>>> True
"rage" in teacher
list(teacher)
```

Some basic operations on dictionaries

```
teacher = {
   "firstname": "John", "lastname": "Wick", "age": 52,
   "address": {
       "number": 34,"street": "rue Saint Bernard",
       "postcode": 57910,"town": "Berkange"},
   "courses": ["programming", "ai"]
}
type(teacher) #>>> dict
len(teacher) #>>> 5
len(teacher["address"]) #>>> 4
"age" in teacher #>>> True
"rage" in teacher #>>> False
list(teacher)
```

Some basic operations on dictionaries

```
teacher = {
   "firstname": "John", "lastname": "Wick", "age": 52,
   "address": {
       "number": 34,"street": "rue Saint Bernard",
       "postcode": 57910,"town": "Berkange"},
   "courses": ["programming", "ai"]
}
type(teacher) #>>> dict
len(teacher) #>>> 5
len(teacher["address"]) #>>> 4
"age" in teacher #>>> True
"rage" in teacher #>>> False
list(teacher) #>>> ['firstname', 'lastname', 'age', 'address', 'courses']
```

Creating a dictionary step by step

```
 1  student = {}
 2  student["firstname"] = "Alix"
 3  student["lastname"] = "Block"
 4  student["age"] = 20
 5  student["address"] = {}
 6  student["address"]["city"] = "Nancy"
 7  student["address"]["street"] = "place stan"
 8  student["address"]["number"] = 16
 9  student["address"]["postalcode"] = 54000
10  student["courses"] = []
11  student["courses"].append("programming")
12  student["courses"].append("ai")
13  print(student)
#>>> {'firstname': 'Alix', 'lastname': 'Block', 'age': 20, 'address': {'city': 'Nancy', 'street': 'place
stan', 'number': 16, 'postalcode': 54000}, 'courses': ['programming', 'ai']}
```

Iterating over dictionaries

```
student = {
   "firstname": "Alix", "lastname": "Block", "age": 20,
   "address": {
      "city": “Nancy”,"street": "place stan", “number”:16
      "Postalcode”: 54000
   },
   "courses": ["programming", "ai"]
}
for a in student:
   print(a)
```

Iterating over dictionaries

```
student = {
    "firstname": "Alix", "lastname": "Block", "age": 20,
    "address": {
        "city": “Nancy”,"street": "place stan", “number”:16
        "Postalcode”: 54000
    },
    "courses": ["programming", "ai"]
}
for a in student:
    print(a)
```

Creating a dictionary at once

firstname
lastname
age
address
courses

a is the key name

Iterating over dictionaries

```
student = {
   "firstname": "Alix", "lastname": "Block", "age": 20,
   "address": {
       "city": “Nancy”,"street": "place stan", “number”:16
       "Postalcode”: 54000
   },
   "courses": ["programming", "ai"]
}
for a in student:
   print(f'The value of {a} is {student[a]}.')
```

Iterating over dictionaries

```
student = {
   "firstname": "Alix", "lastname": "Block", "age": 20,
   "address": {
      "city": “Nancy”,"street": "place stan", “number”:16
      "Postalcode”: 54000
   },
   "courses": ["programming", "ai"]
}
for a in student:
   print(f'The value of {a} is {student[a]}.')
```

The value of firstname is Alix.
The value of lastname is Block.
The value of age is 20.
The value of address is {'city': 'Nancy', 'street': 'place stan', 'number': 16, 'postalcode': 54000}.
The value of courses is ['programming', 'ai'].

Iterating over dictionaries : by key (k) and value (v)

```
student = {
   "firstname": "Alix", "lastname": "Block", "age": 20,
   "address": {
       "city": “Nancy”,"street": "place stan", “number”:16
       "Postalcode”: 54000
   },
   "courses": ["programming", "ai"]
}
for k, v in student.items():
   print(f'The value of {k} is {v}.')
```

Iterating over dictionaries : by key (k) and value (v)

```
student = {
   "firstname": "Alix", "lastname": "Block", "age": 20,
   "address": {
       "city": “Nancy”,"street": "place stan", “number”:16
       "Postalcode”: 54000
   },
   "courses": ["programming", "ai"]
}
for k, v in student.items():
   print(f'The value of {k} is {v}.')
```

The value of firstname is Alix.
The value of lastname is Block.
The value of age is 20.
The value of address is {'city': 'Nancy', 'street': 'place stan', 'number': 16, 'postalcode': 54000}.
The value of courses is ['programming', 'ai'].

To be seen in labs

Dictionaries.

Game skeleton with dictionaries!