Python Programming

Fonctions

Jean-Christophe Taveau
Patricia Thébault

CPBx 2016-2017

Langage de Programmation: Les bases

→ Les fonctions: Utilisation

→ Les méthodes des objets: Utilisation

→ Les fonctions à faire soi-même

Les Bases: Utilisation des Fonctions

<name> (arg0, arg1,..., argn)

```
result = myFunction(1,2,'cool')
output = otherFunction()
```

```
length0 = len('Supercalifragilisticexpialidocious')
print(length0) # ← 34
length1 = len([0,2,4,6,7])
print(length1) # ← 5

list = range(1,10,2)
print(list) # ← [1,3,5,7,9]

a_string = str(100) # a_string ← '100'
```

Nom des fonctions:
→ Faire attention à la casse. cos(..) et COS(..) différents.
Arguments:
→ entre parenthèses et séparés par des virgules.
→ On peut avoir un mélange de plusieurs types dans les arguments (cf: exemple).
→ La fonction retourne (le plus souvent) une valeur comme en mathématiques f(x) = 2*x + 3; f(2) = 7
→ Une fonction sans arguments doit être appelée avec des parenthèses.

Les Bases: Utilisation des Fonctions

<name> (arguments par défaut)

Exemple: Fonction range
La fonction **range** a 3 arguments:
→ l'initialisation: valeur entière initiale (par défaut, 0)
→ la valeur entière de sortie (argument obligatoire)
→ l'incrément (par défaut +1)

```
list0 = range(7) # list0 ← [0, 1, 2, 3, 4, 5, 6]

list1= range(1,5) # list1 ← [1,2,3,4]

list2 = range(1,10,2) # list2 ← [1,3,5,7,9]
```

Les Bases: Utilisation des Fonctions - Documentation

Le site officiel du langage Python est: http://www.python.org où se trouve la doc.
Pour chaque fonction, on trouve sa syntaxe, son fonctionnement et les contraintes (par ex: nombres entiers)
Les arguments entre crochets signifient qu'ils sont optionnels.

The range type represents an immutable sequence of numbers and is commonly used for looping a specific number of times in for loops.

class **range**(*stop*)
class **range**(*start*, *stop*[, *step*])

The arguments to the range constructor must be integers (either built-in int or any object that implements the __index__ special method). If the *step* argument is omitted, it defaults to 1. If the *start* argument is omitted, it defaults to 0. If *step* is zero, ValueError is raised.

For a positive *step*, the contents of a range r are determined by the formula
r[i] = start + step*i where i >= 0 and r[i] < stop.

For a negative *step*, the contents of the range are still determined by the formula
r[i] = start + step*i, but the constraints are i >= 0 and r[i] > stop.

Langage de Programmation: Les bases

→ Les fonctions: Utilisation

→ Les méthodes des objets: Utilisation

→ Les fonctions à faire soi-même

Les Bases: Les méthodes des objets

<objet>.<method> (arguments)

```
list0 = [1,2,3]
list0.append(4)
print(list0) # ← [1,2,3,4]
```

```
list1 = [1,2,3,4]
list1.reverse()
print(list1) # ← [4,3,2,1]
```

```
list2 = [1,2,3,4,5]
list2.pop()
print(list2) # ← [1,2,3,4]
```

```
dico = ['A':120, 'C':24, 'G': -45]
for key,val in dico.items():
   print(k,v)
```

```
mystr = 'python is good'
words = mystr.split(' ')
print(words)
# ← ['python', 'is',
'good']
```

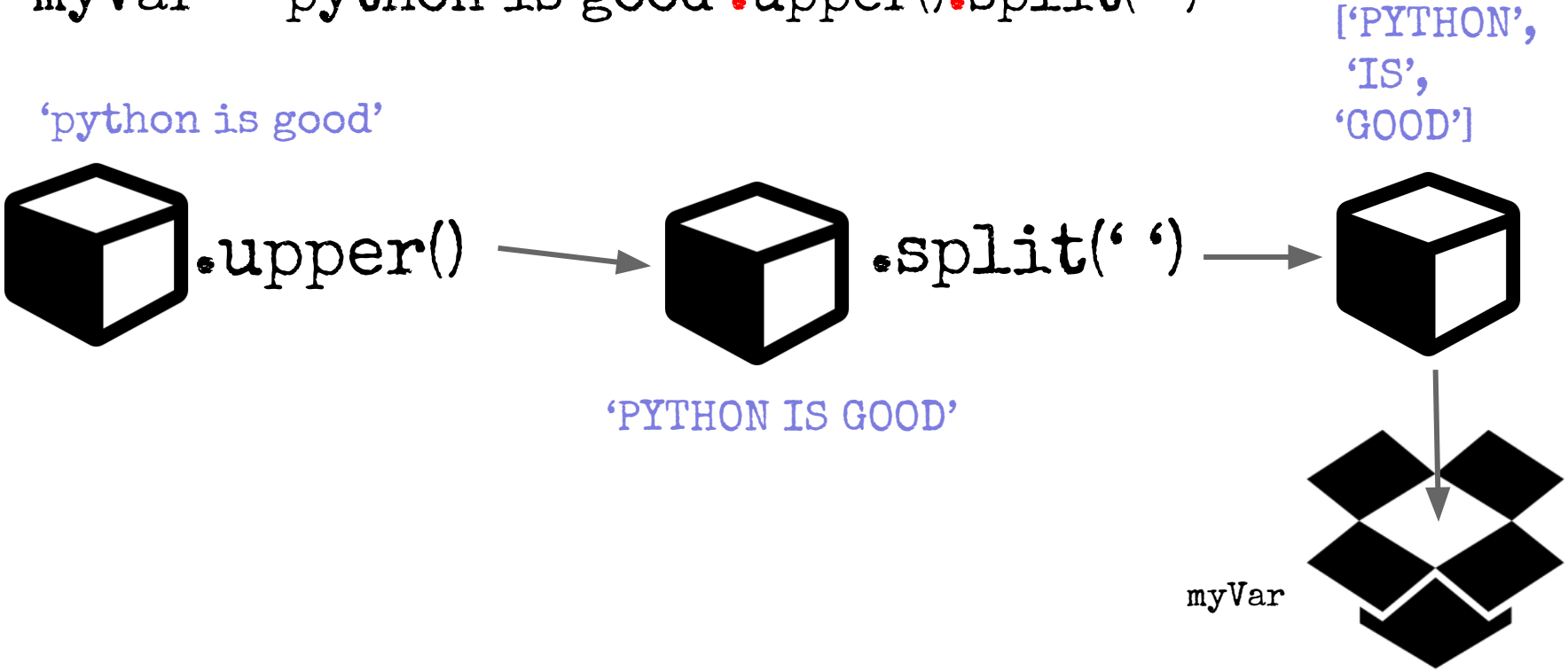
```
seq = 'PYTHCN'
res = seq.lower()
print(res) # ← 'pythcn'
```

Les Bases: Quelques méthodes de List

list.append(*x*)	Add an item to the end of the list. Equivalent to a[len(a):] = [x].
list.insert(*i*, *x*)	Insert an item at a given position. The first argument is the index of the element before which to insert.
list.remove(*x*)	Remove the first item from the list whose value is *x*. It is an error if there is no such item.
list.pop([*i*])	Remove the item at the given position in the list, and return it. If no index is specified, a.pop() removes and returns the last item in the list. (The square brackets around the *i* denote that the parameter is optional)
list.clear()	Remove all items from the list. Equivalent to del a[:].
list.index(*x*)	Return the index in the list of the first item whose value is *x*. It is an error if there is no such item.
list.count(*x*)	Return the number of times *x* appears in the list.
list.sort(...)	Sort the items of the list in place.
list.reverse()	Reverse the elements of the list in place.
list.copy()	Return a shallow copy of the list. Equivalent to a[:].

Les Bases: Chaînage de méthodes

```
myVar = 'python is good'.upper().split(' ')
```



Langage de Programmation: Les bases

→ Les fonctions: Utilisation

→ Les méthodes des objets: Utilisation

→ Les fonctions à faire soi-même

Les Bases: Création/Définition d'une fonction

```
def <name>(<args>) :
    """Description"""
    Bloc d'instructions
    return myResult
```

Nom des fonctions:
→ Même contraintes que les variables.
Arguments:
→ entre parenthèses et séparés par des virgules.
→ On peut avoir un mélange de plusieurs types dans les arguments (cf: exemple).
→ La fonction retourne (le plus souvent) une valeur comme en mathématiques
f(x) = 2*x + 3; f(2) = 7
→ Une fonction sans arguments doit être définie avec des parenthèses.

Les Bases: Création/Définition d'une fonction

```
def add(a,b):
    """Add two numbers"""
    return a+b


myVar = add(2,3)
print(myVar) # ← 5
```