Python
Expressions Lambda

Nicolas Delestre

INSA
ROUEN NORMANDIE

Définition

- Une expression lambda est une fonction anonyme avec une seule instruction qui est une unique expression
  - Non utilisation du mot clé `return`, la valeur de l'expression est la valeur retournée
- Elle est utilisée comme :
  - valeur par défaut d'un paramètre formel
  - paramètre effectif
  - exceptionnellement comme valeur d'une affectation
- Elle évite de créer des fonctions à usage unique

INSA ROUEN NORMANDIE

Syntaxe

```
lambda [param1, param2, ...] : expression
```

Équivalence

```
foo = lambda parametres : expression
```

est équivalent à

```
def foo(parametres):
    return expression
```

INSA ROUEN NORMANDIE

Exemple 1 : ordre de tri

```
def trier(l, doivent_etre_echanges=lambda x, y: x > y):
    est_trie = False
    while not est_trie:
        est_trie = True
        for i in range(len(l)-1):
            if doivent_etre_echanges(l[i], l[i+1]):
                l[i], l[i+1] = l[i+1], l[i]
                est_trie = False
```

```
>>> l=[3,2,8,1,5,98,3]
>>> trier(l)
>>> l
>>> [1, 2, 3, 3, 5, 8, 98]
>>> trier(l,lambda x,y : x < y)
>>> l
>>> [98, 8, 5, 3, 3, 2, 1]
```

Exemple 2 : généralisation du tri 1 / 2

```
def trier(l,
          doivent_etre_echanges=lambda x, y: x > y,
          valeur_a_comparer=lambda x: x):
    est_trie = False
    while not est_trie:
        est_trie = True
        for i in range(len(l)-1):
            if doivent_etre_echanges(valeur_a_comparer(l[i]),
                                     valeur_a_comparer(l[i+1])):
                l[i], l[i+1] = l[i+1], l[i]
                est_trie = False
```

INSA ROUEN NORMANDIE

Exemple 2 : généralisation du tri 2 / 2

```
>>> l=[complex(3,1), complex(1,3), complex(2,2)]
>>> trier(l)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "<stdin>", line 8, in trier
  File "<stdin>", line 2, in <lambda>
TypeError: '>' not supported between instances of 'complex' and 'complex'
>>> trier(l, valeur_a_comparer=lambda z:z.real)
>>> l
[(1+3j), (2+2j), (3+1j)]
>>> trier(l, valeur_a_comparer=lambda z:z.imag)
>>> l
[(3+1j), (2+2j), (1+3j)]
>>> from math import sqrt
>>> trier(l, valeur_a_comparer=lambda z:sqrt(z.real**2+z.imag**2))
>>> l
[(2+2j), (3+1j), (1+3j)]
```

INSA ROUEN NORMANDIE

Conclusion

Les expressions lambda :

- sont des fonctions anonymes très simples
- sont issues de la programmation fonctionnelle
- permettent de généraliser des algorithmes
- permettent d'éviter de créer des fonctions peu utiles

INSA ROUEN NORMANDIE