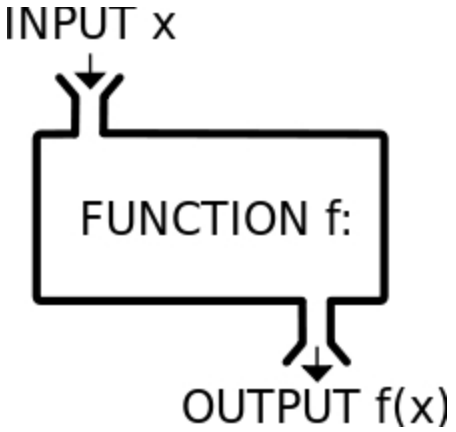
TP - Programmation fonctionnelle



Cas pratique

par **Fabien Barbaud** - **@BarbaudFabien**

Exercice 1

Convertir en programmation fonctionnelle - *reduce*

```
numbers = [1, 2, 3, 4]
total = 0
for number in numbers:
  total += number
print(total)
```

Réponse

```
import functools
numbers = [1, 2, 3, 4]
functools.reduce(lambda x, y: x + y, numbers)
```

```
(((1+2)+3)+4)
```

Exercice 2

Convertir en programmation fonctionnelle - *map*

```
items = [1, 2, 3, 4, 5]
squared = []
for x in items:
  squared.append(x ** 2)
print(squared)
```

Réponse

```
items = [1, 2, 3, 4, 5]
list(map(lambda x: x ** 2, items))
```

Exercice 3

Convertir en programmation fonctionnelle - *filter*

```
result = []
for x in range(-5, 5):
  if x < 0:
    result.append(x)
print(result)
```

Réponse

```
list(filter(lambda x: x < 0, range(-5,5)))
```

Exercice 4

Convertir en programmation fonctionnelle - *reduce*

```
L = ['Chef ', 'de ', 'projet ', 'digital']
''.join(L)
```

```
'Chef de projet digital'
```

Réponse

```
import functools
L = ['Chef ', 'de ', 'projet ', 'digital']
functools.reduce( (lambda x,y:x+y), L)
```

Exercice 5

Faire la somme du tableau suivant

```
chiffres = ['un', 'deux', 'trois', 'quatre']
```

Réponse

```
import functools
chiffres = ['un', 'deux', 'trois', 'quatre']

def convertion(chiffre):
  if chiffre == 'un':
    return 1
  elif chiffre == 'deux':
    return 2
  elif chiffre == 'trois':
    return 3
  elif chiffre == 'quatre':
    return 4

vraiChiffres = map(convertion, chiffres)
functools.reduce(lambda x,y:x+y, vraiChiffres)
```

Réponse (bis)

```
import functools
chiffres = ['un', 'deux', 'trois', 'quatre']

def convertion(chiffre):
  dico = {"un": 1, "deux": 2, "trois": 3, "quatre": 4}
  return dico[chiffre]

vraiChiffres = map(convertion, chiffres)
functools.reduce(lambda x,y:x+y, vraiChiffres)
```