Introduction à Python II

Chuan Xu

chuan.xu@univ-cotedazur.fr

Sept. 2024

Modules

- Stockez vos fonctions dans un fichier python séparé appelé **module** and puis **import** ce module dans votre programme principal
- Permet de réutiliser des fonctions dans de nombreux programmes différents

```
import math # Module standard
math.sqrt(4) # Appeler la fonction dans le module math
# L'appel des fonctions se fait via le nom du module

import timeit as t # Donne un alias au module importé
t.timeit("[x**2 for x in range(1_000_000)]", number=10)
# Mesurer le temps en seconde
```

Packages

- Packages : les modules organisés en une arborescence de répertoires

```
sound/
      __init__.py
      formats/
              __init__.py
              wavread.py
              wavwrite.py
              aiffread.py
              aiffwrite.py
              auread.py
              auwrite.py
              ...
      effects/
              __init__.py
              echo.py
              surround.py
              reverse.py
              ...
```

```python
# Importe sous-modules individuellement
import sound.effects.echo
sound.effects.echo.echofilter(input, output)

# L'autre manière: from package import element
from sound.effects import echo
echo.echofilter(input, output)

# Importe directement la fonction
from sound.effects.echo import echofilter
echofilter(input, output)
```

Références interne dans un package

Module surround.py

```
sound/
      __init__.py
      formats/
              __init__.py
              wavread.py
              wavwrite.py
              aiffread.py
              aiffwrite.py
              auread.py
              auwrite.py
              ...
      effects/
              __init__.py
              echo.py
              surround.py
              reverse.py
              ...
```

```
from . import echo
from .. import formats
from ..formats import wavread
```

Introduction à packages numpy

- L'objectif principal de **numpy** est de fournir une structure de données très efficace appelée *tableau numpy* (*tenseurs*), et les outils pour manipuler ces tableaux.
- Numpy n'est pas un module standard, il faut l'installer `conda install numpy`
- Numpy est rapide avec les codes optimisé pour le CPU

```
import timeit as t # Donne un alias au module importé
t.timeit("[x**2 for x in range(1_000_000)]", number=10)

import numpy as np
t.timeit("np.array(range(1_000_000))**2", globals=globals(), number=10)
```

Introduction à la bibliothèque NumPy

- L'objectif principal de **numpy** est de fournir une structure de données très efficace appelée *tableau numpy* (*tenseurs*), et les outils pour manipuler ces tableaux.
- Numpy n'est pas un module standard, il faut l'installer `conda install numpy`
- Numpy est rapide avec les codes optimisé pour le CPU

```
import timeit as t # Donne un alias au module importé
t.timeit("[x**2 for x in range(1_000_000)]", number=10)

import numpy as np
t.timeit("np.array(range(1_000_000))**2", globals=globals(), number=10)
```

Création de tableaux numpy

```
import numpy as np
#Créer des tableaux avec différent types
x = np.array([0, 1, 2])
print(x.dtype)

x = np.array([0., 1., 2.])
print(x.dtype)

x = np.array([0., 1., 2.], dtype=int)
print(x.dtype)

x = np.array([0, 1, 2], dtype=np.int8)
print(x.dtype)

x = np.zeros((2,3)) #Créer une matrice de 0
y = np.ones((2,3,4,5)) #Créer un tenseur de 1 avec 4 dimension
print(x.shape, x.shape[0], x.shape[1], x.ndim, y.ndim)
print(x[0,1]) # Accéder au élément de tableau

x = np.random.rand(2,3) # Créer une matrice aléaratoire de valeur [0,1)
x = np.arange(10)+1 # Créer un vecteur de 1 à 10
```

Les bases numpy - Vue et copie d'un tableau

```
import numpy as np

x = np.arange(10)+1 # Créer un vecteur de 1 à 10
a = x.reshape(2,5) # Redimensionner le tenseur, renvoie une vue du tableau
x[0] = 0 # Quand la valeur initial change
a
a[0,0] = 1 # Quand la valeur de tenseur redimensionné change
x

a = x.reshape(2,5).copy() # renvoie d'une copie de tableau
x[0]=0
a



b = a[:,1] # Sélectionner la deuxième colonne, renvoie une vue du tableau
c = a[:,:2] # Sélectionner les deux premières colonnes
d = a[1, :] # Séclectionner la deuxième ligne
at = a.transpose() # Le transpose d'un tableau, renvoie une vue du tableau
```

Les bases numpy - Aplatir et concaténer

```
import numpy as np

x = np.arange(10)+1 # Créer un vecteur de 1 à 10
a = x.reshape(2,5)
e = a.flatten() # Aplatir un tableau, renvoie une copie du tableau

y = np.arange(10)+11
v = np.c_[x, y] # Concaténer deux tableaux sur le dernier axe
```

Opération numpy

Faire une opération terme à terme de même dimension.

```
import numpy as np

x = np.arange(9)+1 # Créer un vecteur de 1 à 9
y = x**2 # Calculer pour tous les éléments à la puissance 2.
z = np.exp(x) # Calculer l'exponential de tous les éléments

a = x.reshape(3,3)
b = a+10
c = a*b # multiplier terme à terme deux tableaux !
d = np.dot(a,b) # Le produit matriciel
d1 = a @ b # Le produit matriciel aussi

e = np.sum(a) # Retourne la somme totale de la matrice a
f = np.sum(a, axis=0) # Retourne les sommes des colonnes
g = np.sum(a, axis=1) # Retourne les sommes des lignes

e = np.mean(a) # Retourne le moyenne de tous les éléments dans la matrice a
```

Booléens

```
import numpy as np

x = np.arange(9)+1 # Créer un vecteur de 1 à 9
y = x<3 # Renvoie un tableau de taille égale à celle de x
# dont les éléments sont des booléens et correspondent au test < terme à terme
```

Temps d'exercices !
MOODLE TP: Utilisation de python basique II