Programmer avec Python
Cours

Séance n° 5	**Les modules**

Lors de la première séance, nous avons revu certaines fonctions intégrées à Python comme la fonction `print` ou la fonction `input`. Bien sûr, toutes les fonctions ne sont pas intégrées à Python, nous avons d'ailleurs revu comment écrire nos propres fonctions. Seules les fonctions les plus utilisées sont intégrées à Python : en effet afficher un message à l'écran, récupérer une saisie clavier ou calculer la longueur d'une chaîne avec `len` sont des opérations très courantes dans un programme. Pour les usages moins courants, nous avons à notre disposition des **modules**. Il s'agit de fichiers regroupant des classes ou des fonctions, le plus souvent partageant une même thématique. Un grand nombre de modules sont fournis avec Python. Par exemple le module `math` contient de nombreuses fonctions mathématiques. Nous allons voir quelques exemples et comment les utiliser.

1 Les modules intégrés à Python

1.1 Le module `math`

Importer l'intégralité d'un module

Nous l'avons dit, le module `math` contient de nombreuses fonctions mathématiques comme le cosinus ou la racine carrée. Pour l'utiliser, il suffit d'écrire cette ligne au tout début de votre programme :

```
import math
```

Vous pourrez alors faire usage de tout le contenu du module : toutes les fonctions mais aussi par exemple, la variable `pi`. Un module peut tout à fait contenir également des variables ! Si la liste de toutes les fonctions et variables du module `math` vous intéresse, vous pouvez consulter la documentation du module à cette adresse : https://docs.python.org/3/library/math.html.

Pour utiliser un élément du module, il faut suivre cette syntaxe :

```
nom_du_module.nom_element
```

Par exemple :

```
print(math.factorial(5)) # Affiche la factorielle de 5
```

```
print(math.pi) # J'affiche la valeur de pi
```

Voyons maintenant un exemple complet :

```
import math

angleDegres = 90 # Un angle en degrés
angleRadians = math.radians(angleDegres) # On convertit en radians

print(angleRadians) # On affiche

print(math.sin(angleRadians)) # On calcule le sinus de l'angle et
    on affiche
```

Cet exemple utilise la variable `pi` du module math et les fonctions `radians` pour convertir un angle en degrés en radians et `sin` pour calculer le sinus.

Importer directement une fonction

Parfois, vous ne voulez pas importer tout un module en entier mais seulement certains éléments. Il est alors possible de les importer directement et vous n'aurez pas alors besoin de préfixer l'élément par son nom de module quand vous l'utiliserez.

```
from math import factorial # On importe directement la fonction
   factorial de math

f = factorial(5) # Pas besoin d'utiliser le préfixe math.
print(f)
```

Si vous souhaitez importer plusieurs éléments d'un même module, vous pouvez utiliser une seule instruction :

```
from math import factorial, sin, cos
```

Évitez d'utiliser la syntaxe `from module import *` que vous verrez souvent sur le net. C'est une mauvaise pratique. Elle a pour effet d'importer tous les éléments d'un module et de pouvoir les utiliser sans préfixe. Cela semble bien pratique certes mais vous ignorez ce que vous importez exactement et alors gare aux confusions de noms. Imaginez un programme où vous utilisez la variable `e` un peu partout, et un beau jour vous décidez d'importer tout le module math directement avec `from math import *`. Et bien vous aurez un problème puisque la variable `e` existe aussi dans le module `math` !

Les alias

Vous me direz alors que les modules ont l'air bien pratique mais que devoir écrire systématiquement leur nom est fatiguant. Une solution à ça : les alias. Vous pouvez donner un alias à un nom de module pour aller plus vite. Exemple :

```
import math as m #On pourra utiliser m plutôt que math dans tout le
    programme

nombre = m.factorial(5)
print(nombre)
```

1.2 Le module `datetime`

Comme son nom l'indique, le module `datetime` fournit des classes pour manipuler des dates et heures.

- La classe `date` comporte les attributs `year`, `month` et `day`
- La classe `time` comporte les attributs `hour`, `minute`, `second`, `microsecond`, et `tzinfo` (pour timezone)
- La classe `datetime` qui combine les attributs des deux premières classes
- La classe `timedelta` qui permet d'exprimer la différence entre deux temps, en microsecondes

Obtenir la date courante

```
from datetime import date

today = date.today()

print("Nous sommes le", today)
```

Créer une date

```
from datetime import date

rentree = date(2022, 9, 8) # année, mois, jour !

print("La date créée est", rentree)
```

Obtenir l'année, le mois ou le jour à partir d'une date

```
from datetime import date

today = date.today()

print("Année :", today.year)
print("Mois :", today.month)
print("Jour :", today.day)
```

Créer une heure

```
from datetime import time

horloge = time(13, 24, 56) # heures, minutes, secondes

print("L'heure est ", horloge)
```

Obtenir les heures, minutes, secondes

```
from datetime import time

horloge = time(11, 55, 45)

print("hour =", horloge.hour)
print("minute =", horloge.minute)
print("second =", horloge.second)
print("microsecond =", horloge.microsecond)
```

Obtenir l'heure courante

Pour obtenir l'heure, vous devrez utiliser la classe `datetime` qui permet d'utiliser à la fois les attributs pythoninlineyear, `month`, `day` et pythoninlinehour, `minute`, `second`, `microsecond`, `tzinfo`.

```
from datetime import datetime

today = datetime.now()

print("Date et heure", today)
print("Juste l'heure", today.hour)
print("L'heure et les minutes", today.hour, today.minute)
```

Formatter une date/heure

```
from datetime import datetime

# Récupérer l'heure courante
now = datetime.now()


# Date type Jeudi 8 septembre 2022
s = now.strftime("%A %d %B %Y")
print("Exemple :", s)

# Heure type heures:minutes:secondes
s = now.strftime("%H:%M:%S")
print("Exemple :", s)
```

Pour voir la liste complète des possibilités de formatage, voir https://docs.python.org/fr/3.6/library/datetime.html#strftime-strptime-behavior.

Obtenir la durée entre deux dates

```
from datetime import datetime, timedelta

# L'heure courante
now = datetime.now()
print ("En ce moment = ", now)

# on calcule la date-heure dans 2 jours
surlendemain = now + timedelta(days = 2)

# on calcule la date-heure il y a 12 heures
avant = now - timedelta(hours = 12)

# on affiche
print ("Dans 2 jours on sera", surlendemain)
print("Il y a 12 heures on était", avant)

# On calcule la durée entre en ce moment et dans deux jours
duree = now-surlendemain
print("La durée jusqu'au surlendemain c'est", duree)
```

Vous pouvez ainsi additionner des dates ou les soustraire mais aussi les multiplier et les diviser.

1.3 Et beaucoup d'autres !

Python contient bien d'autres modules que vous pouvez importer et utiliser de la même façon que celle que nous venons de voir avec le module `math` :

- Le module os fournit une manière portable dutiliser les fonctionnalités dépendantes du système dexploitation
- Le module random (que nous avons vu en première année !) permet de générer des nombres aléatoires
- Le module sys fournit un accès à certaines variables système
- Le module numpy pour le calcul scientifique

— Le module matplotlib permet de créer des graphiques
— Le module scikit-learn regroupe des outils pour réaliser des analyses prédictives de données
— Le module pillow pour manipuler des images
— Et de nombreux modules dédiés à l'intelligence articicielle comme pytorch, tensor flow ou encore keras.

2 Créer ses propres modules

Tout fichier Python portant l'extension .py peut être utilisé comme un module. Vous pouvez ainsi découper un trop long programme en plusieurs fichiers. Un fichier principal servira pour le lancement du programme et les classes ou fonctions nécessaires seront déportées dans des modules.

Reprenons par exemple la correction de l'exercice de la séance 1 où vous deviez écrire un programme et une fonction calculant la FCmax d'un utilisateur :

```
# La fonction fcmax prend en entrée un âge et qui renvoie
# en sortie la valeur de FCM estimée par cette formule.
def fcmax(age):
    return 192-0.007 * age * age

# Programme principal
age = float(input("Saisissez votre âge en années : "))

calcul = fcmax(age)
print("Votre fréqunce cardiaque maximale estimée est", calcul)
```

Découpons ce programme en un fichier principal et un module.
Le fichier `sport.py` comportera notre fonction de calcul :

```
# La fonction fcmax prend en entrée un âge et qui renvoie
# en sortie la valeur de FCM estimée par cette formule.
def fcmax(age):
    return 192-0.007 * age * age
```

Le fichier principal `test.py` comportera le programme principal uniquement et on oubliera pas d'ajouter la ligne permettant d'important notre nouveau module `sport` ainsi que le préfixe devant la fonction :

```
import sport

# Programme principal
age = float(input("Saisissez votre âge en années : "))

calcul = sport.fcmax(age)
print("Votre fréqunce cardiaque maximale estimée est", calcul)
```

Et voilà, nous avons créé notre propre module Python `sport` !