

CHAPITRE IV : Modules, mise au point

Présentation

Python possède certaines fonctions de bases prédéfinies comme `input()`, `print()`, `range()`, ..
 Mais selon les sujets étudiés, d'autres fonctions plus spécialisées s'avéreront nécessaires.
 Elles sont regroupées en bibliothèques nommées **modules** : `math` (pour les calculs scientifiques),
`random` (gestion du hasard), `turtle` (dessins), `tkinter` (Interface), `time` (gestion du temps), etc ...

Il faudra les **importer** au cas par cas selon les besoins

Importation d'un module : retenez ces deux méthodes.

import *module* , puis appel d'une fonction : *module.fonc(...)*
 ou
from *module* **import** * , puis appel d'une fonction : **fonc(...)**

NB : Pour connaître le détail des fonctions d'un module on tape **help("module")** dans la console python.

Conseils d'usage

« `from module import *` » convient à un programme court et spécialisé. Le script sera plus allégé.
 « `import module` » convient à un projet plus complexe ou général pour éviter les conflits entre noms
 fonctions identiques. Enfin on peut importer un module d'une manière, et un second d'une autre.
 On écrira l'une de ces instructions au tout début, (première ou deuxième ligne).

Ex1 : Module math

Dans le Shell, tapez `help("math")`

1. Compléter la documentation ci-contre par les noms des instructions Python trouvées avec `help` en console.
2. Tester ces fonctions dans le Shell sans oublier d'importer le module `math` avant de tester...

Quelques fonctions Python du module **math**

racine carré d'un nombre x :
 valeur absolue d'un nombre x :
 pgcd de 2 nombres x et y :
 convertir radians en degrés :
 nom de la variable π :

3. a) Faire un programme donnant la longueur de l'hypoténuse d'un triangle ABC rectangle en A, après avoir demandé les longueurs des autres cotés.
 b) Faire calculer et afficher les deux angles \hat{B} et \hat{C} au degré près.
 c) Afficher l'aire et le périmètre du cercle circonscrit.

Ex2 : Module random

Dans le Shell, tapez `help("random")`
 et compléter le tableau ci-contre :

Quelques fonctions Python du module **random**

randint (a,b)	
random ()	

Tester ces fonctions dans le Shell.

Faire un programme simulant 20 lancers d'un dé à six faces comme sur l'exemple ci-contre.

```
>>>
lancer 1 : 4
lancer 2 : 3
lancer 3 : 2
lancer 4 : 6
```

Ex 3 : Module time

Dans le Shell, tapez `help("time")` et compléter le tableau ci-dessous :

<code>sleep(t)</code>	
<code>time()</code>	
<code>localtime()</code>	
<code>strftime("%H')</code> avec %M ou %S : avec %d ou %m ou %Y :	

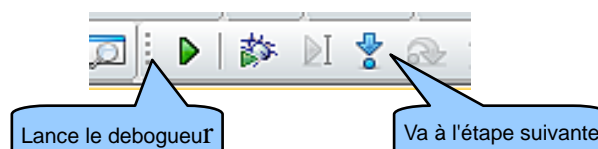
Tester ces fonctions dans le Shell.

1. Faire un script qui affiche un compte à rebours des secondes à partir de 10, puis ' Prêt '
2. Faire un script qui demande votre année de naissance. Ensuite il doit afficher ' On est le.....à.....' avec la date et l'heure d'aujourd'hui, puis doit afficher 'Vous avez.....ans ' avec votre âge .

Mise au point d'un programme

Le bug n'est pas un accident mais plutôt un état normal d'un système informatique et le débogage est donc une phase incontournable. Voici un premier bilan des outils à votre disposition.

- **Commenter et documenter.** Agit préventivement en facilitant la relecture.
- **Tester votre programme.** Choisir des cas particuliers (0, rien , ...), ou dont le résultat est connu.
- **Instrumenter votre programme.** C'est à dire parsemer le programme d'« affichage test » de variables ciblées. A effacer ou désactiver ensuite (avec #) . Simple mais efficace.
- **Utiliser le débogueur.** Les éditeurs possèdent toujours ce type d'outil comme ici avec Pyscripter.
Première approche : tenter une exécution pas à pas pour tenter le cerner le problème.



Ex 4 : Premier jeu (ter)

Reprendre l'Ex10 du chapitre III en donnant des valeurs aléatoires aux coordonnées du bateau.

Ex 5 : Juste prix (ter)

Reprendre l'Ex10 du chapitre III en donnant une valeur aléatoire au prix à trouver.

- Ajouter un code triche.
- Faire des niveaux de difficultés croissantes.

Ex 6 : Le jeu des allumettes

Vous connaissez sûrement le jeu des allumettes, peut-être sous un autre nom. Ce jeu se joue à 2 joueurs, voici son principe :

On dispose sur la table un petit tas d'allumettes. Chaque joueur prend à tour de rôle soit 1, soit 2, soit 3 allumettes. Le joueur qui prend la dernière allumette a perdu. Pour gagner, il faut donc laisser la dernière allumette à son adversaire.

Ecrire un programme de ce jeu ou vous jouerez contre l'ordinateur, l'ordinateur choisissant aléatoirement le nombre d'allumettes à retirer. Afficher après chaque tour, le nombre d'allumettes restantes.

Amélioration possible : limiter le temps pour la réponse du joueur, afficher les allumettes restantes après chaque tour.