CHAPITRE VI : 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

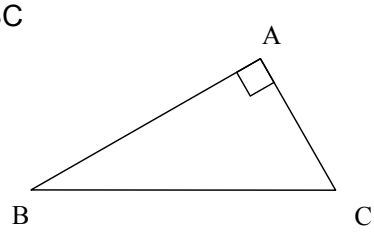
Ouvrir un nouveau script Python

1. Compléter la documentation ci-contre par les noms des instructions Python trouvées avec help en console.

```
from math import *
""" Quelques fonctions Python

racine carré    :
fonctions cos sin et réciproques :
partie entière  :
convertir radians en degrés :
nom de la variable π :
"""
```

2. 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.
   Par exemple si AC=3, AB=4, BC vaudra … ?
   b) Faire calculer et afficher les deux angles $\hat{B}$ et $\hat{C}$ au degré près
   c) Pour les rapides. Afficher l'aire et le périmètre du cercle circonscrit



Ex2 : Module random

Voici deux fonctions du module random

randrange (a,b)	Renvoie un entier aléatoire dans [a ; b[
random ()	Renvoie un flottant dans [0 ; 1[

Faire un programme simulant 20 lancers d'un dé à six faces.

Afficher la liste des résultats.

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

Ex3 : Module turtle (pour les rapides)

Quelques commandes usuelles

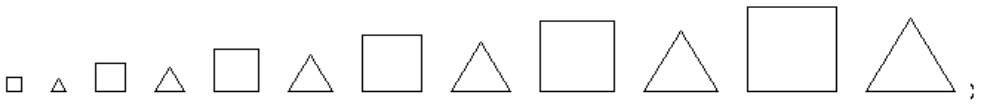
Terminer le script par mainloop()

reset()	*Effacer tout et recommencer*
goto(x,y)	*Aller à l'endroit (x,y)*
forward(x)	*Avancer de la distance x*
backward(x)	*Reculer de la distance x*
up()	*Relever le crayon pour ne plus dessiner*
down()	*Abaisser le crayon pour dessiner*
left(x)	*Tourner à gauche d'un angle x en degrés.*
right(x)	*Tourner à droite d'un angle x en degrés*

1. Faire un script qui dessine un carré de côté 20

2. Le modifier pour ajouter à côté de lui un triangle équilatéral de côté 20

3. Compléter le script pour réaliser une série de carrés et triangles, de côtés croissants comme ici.



Ex4 : Module time (pour les rapides)

sleep(t)	*Effectue une pause de t secondes*
time()	***Renvoie la date POSIX*** *(temps écoulé depuis 01/01/1970 en secondes).*
localtime( )	***Renvoie la structure de temps*** *locale, actuelle.*
*localtime()**.tm_day*** *(ou tm_mon, tm_year, tm_hour, tm_min, tm_sec)*	***Renvoie la variable numérique 'jour' de la structure*** *de temps* *(ou le mois, l'année, l'heure, les minutes, les secondes)*
strftime(*'Il est* ***%H*** *heure ', localtime())* *(et aussi : %H %M %S %d %m %Y)*	***Renvoie une chaîne de caractère ,*** *intégrant les éléments de la structure de temps précédés de % . (Heure, Minutes, secondes, jours, mois, année,..)*

Préambule : observez ce que donnent time() et localtime() avec la console de l'interpréteur.

1. Faire un script qui affiche un compte à rebours des secondes à partir de 5, puis ' Prêt '

2. Compléter le script pour qu'il 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 .

3. Compléter le programme pour qu'il pose une question, puis si la réponse est juste affiche 'Gagné en ….secondes' avec le temps écoulé , sinon affiche 'Perdu'.

Mise au point d'un programme

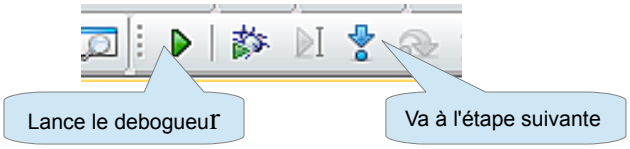
Le bug n'est pas un accident mais plutôt un état normal d'un système informatique et le débogage est donc une phase incontournable. Voici un premier bilan des outils à votre disposition.

- 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.
Idée : déclarer une variable debug=1 et faire précéder les «affichages test» par « if debug: »
Ainsi quand le programme est au point, affecter True à debug désactivera les « affichages tests ».

- 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.



Mettez en application ce que vous avez appris dans l'exercice suivant

Ex5 : Mini projet 'Le juste prix'

1. Écrire un programme qui demande de trouver un prix (entier) compris entre 30 et 100 € , affiche « C'est plus » ou « C'est moins » jusqu'à ce que vous ayez « Gagné ! »

2. Afficher le nombre de coups au final.

3. Limiter à 10 coups, puis donner la réponse et préciser « Perdu ! » si en 10 essais on n'a pas trouvé.

4. Commenter, et faites figurer des tests en fin de code.

Développements au choix pour les rapides :

4. Ajouter un code triche

5. Proposer de rejouer en fin de partie, comptabiliser les parties gagnées, ...

6. Faire des niveaux de difficulté croissante, ….