D) LES VARIABLES EN PYTHON

Les variables rencontrées jusqu'ici sont d'une seule catégorie, celle des entiers (Integer en anglais)

Principaux types de variables

- **INT** et **FLOAT** représentent les **entiers** et les **décimaux**
- **STRING** représente les caractères et les **chaînes de caractères**
- **BOOLEAN** représente les booléens, qui ne peuvent prendre que la valeur **TRUE** ou **FALSE**.
- **LIST** représente les **listes d'éléments divers** : entiers , caractères, ou booléens, ...

D1 : Types de variables. Tester directement en dans la **console d'exécution** les instructions suivantes.

	Deviner le résultat affiché	Vérification, notes
>>>a=3 >>>type(a)	.....	.....
>>>type(a>4)	.....	.....
>>>type(a==3)	.....	.....
>>>b=5.5 >>>type(b)	.....	.....
>>>c=' 47 ' >>>type(c)	.....	.....
>>>d= ' Douze ' >>>type(d)	.....	.....
>>>e=[1,2,a,b, True, 'Salut'] >>>type(e)	.....	.....

D2 : **Opérations et variables.** Devinez puis vérifier les résultats des opérations suivantes

>>>a+b	.....	Additionner deux types de nombres
>>>c+d	.....	On dit **concaténer** les chaînes c et d
>>>print(c+d)	.....	Imprime ce qui est **entre les guillemets**
>>>a+c	.....	Ces deux types ne peuvent s'ajouter
>>> a*2	.....	Multiplier par deux
>>>a**2	.....	Puissance 2
>>>a**3	.....	Puissance 3
>>>c*2	.....	Reproduit la chaîne deux fois
>>>c*3	.....	Reproduit la chaîne trois fois
>>> print(not a==3 , b!=5.5)	.....	C'est de la logique.
>>> print(a==3 and b==5)	.....	Les 2 conditions sont elles réalisées ?

D3 : Le transtypage consiste à traduire en un autre type, le type d'une variable : par exemple un type chaîne en un type numérique. Devinez puis tester par exemple ce qui suit.

>>> print(float (a))	.....	a est un entier mais float(a) est un décimal.
>>> print(str(a)+str(b))	....	«str() » traduit les nombres en caractères
>>> a+int(c)	....	Ici c est un caractère « int(c) » le traduit en entier.

Appeler le professeur pour validation

E) MODULES ET FONCTIONS ASSOCIEES

Il y a un certain nombre de fonctions prédéfinies en Python, comme input(), print(), …, mais nous aurons besoin d'utiliser d'autres fonctions plus variées ou spécialisées.

Elles sont regroupées en **modules** que l'on doit au préalable **importer.**

Modules fréquents : math , random (gestion du hasard), time (gestion du temps), turtle (dessins)

Importation d'un module

from math import * importe tout le module math, et ensuite on tape seulement **fonc(..)**

C'est plus simple pour débuter mais on risque d'avoir plusieurs fonctions de même nom à l'avenir...

import math *importe tout le module math mais il faut taper* **math.fonc(..)** *pour utiliser la fonction fonc(..)*

On peut aussi importer une seule fonction avec « from math import fonc », puis faire fonc()

Taper **help("math")** dans la console python permet de voir toutes les fonctions du module math

Ex E1 : Trigonométrie

Ci-contre les première lignes avec AmiensPython :

La 2ème force la division à être décimale. (inutile en Python3)

La 3ème importe le module **lycee.**

```
# Créé par Leclerc, le 15/06/2013
from __future__ import division
from lycee import *
```

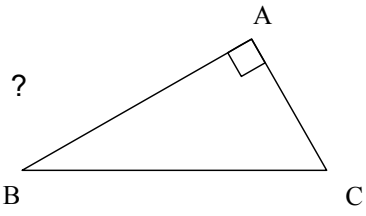
*C'est une compilation des modules **math** et **random**, complétée et commentée en français.*

- Taper ***help("lycee")*** en ligne de commande et retrouver le nom des fonctions ci dessous.

Racine carrée de x	
Les fonctions trigonométriques	
Partie entière de x	
Valeur absolue de x	
Valeur de π	

- Faire un programme calculant de la longueur de l'hypoténuse d'un triangle rectangle : Par exemple ci contre AC=3, AB=4, BC vaudra … ?

- Faire calculer au degré près les deux angles $\hat{B}$ et $\hat{C}$



- Pour les rapides . Calculer le périmètre et l'aire de son cercle circonscrit

Ex E2 : Lancer de dés

Voici deux premières fonctions liées au hasard du module **random**

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

Générez une séquence aléatoire de 20 chiffres, ceux-ci étant des entiers tirés entre 1 et 6 -

Appeler le professeur pour validation

COMMENTAIRES, INSTRUMENTATION, TESTS

Les règles du bon programmeur

Bien écrire un programme

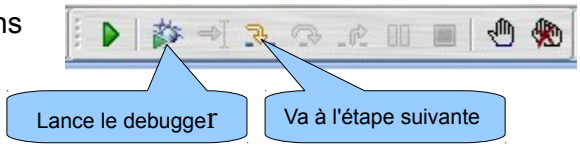
- **Donner des noms évocateurs aux variables :** C'est le premier réflexe à avoir quand on écrit un programme. Par exemple : tirage_1D , longueur_du_cote
- **Commenter vos programmes : #** (« croisillon » ou « dièse »)
  Décrire sur le reste de la ligne ce que vous faites, sans que cela ne soit exécuté.

Même si cela paraît plus long, on y gagne en rapidité de compréhension pour soi même, pour les autres (enseignants, coéquipier), et quand on revient sur son sujet après plusieurs jours ou semaines après.

Mise au point d'un programme : Instrumentation.

- **Tester votre programme avec des cas particuliers** dont on connaît le résultat :
  Faites plusieurs cas selon les sorties possibles, et aussi les cas limites (avec 0, rien , ...)
- **Identifier les variables critiques** (qui ont une influence importante) et **ajouter des affichages** aux endroits clefs pour debugger ou optimiser un programme.
  Vous pourrez toujours désactiver ensuite la ligne avec un #

NB : Les éditeurs possèdent aussi parfois des fonctions de debuggage intégrées, par exemple avec Pyscripter.



Cela ne résout pas le problème mais permet une exécution pas à pas pour tenter de le cerner

Ex E3 : Mini projet 'Le juste prix' .

Appliquez vous à l'écrire et le commenter et le plus clairement possible

1. Écrire un programme qui vous demande de trouver un prix compris entre 1 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 l'ordi donne la réponse et précise « Perdu ! » si en 10 essais on n'a pas trouvé.

En plus pour les rapides :

4. Ajouter un code triche

5. Proposer de rejouer en fin de partie

6. Afficher le nombre moyen de coups par parties, le pourcentage de parties gagnées ….

7. ……..

Éléments de réponses

Pour vérifier , comparer ou si vous êtes bloqués

D1 : Aperçu des types de variables.

```
>>> a=3
>>> type(a)
<type 'int'>        # a est un entier
>>> type(a>4)
<type 'bool'>       # a>4 est un booléen: il est vrai ou faux
>>> type(a==3)
<type 'bool'>       # les booléens résultent des opérations logiques or, not, and
>>> b=5.5
>>> type (b)
<type 'float'>      # b est un décimal à virgule flottante
>>> c='47'
>>> type(c)
<type 'str'>        # c est une chaîne de deux caratères 4 et 7
>>> d=' est un nombre premier'
>>> type(d)
<type 'str'>        # d est une chaîne de quatre mots
>>> e=[1,2,a,b,True,'salut']
>>> type(e)
<type 'list'>       # e est une liste de 6 éléments de quatre types
```

D2 : Opérations et types de variables.

```
>>> a+b
8.5
>>> c+d
'47 est un nombre premier'
>>> print(c+d)
47 est un nombre premier
>>> a+c
Traceback (most recent call last):
  File "<interactive input>", line 1, in <module>
TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

```
>>> a*2
6
>>> a**2
9
>>> a**3
27
>>> c*2
'4747'
>>> c*3
'474747'
>>> print(not a==3,b!=5.5)
(False, False)
>>> print(a==3 and b==5)
False
```

D3 :

```
>>> print(float(a))          donne   3.0
>>> print(str(a)+str(b))     donne  35.5   ( le résultat de 3 et 5.5 concaténés)
>>> a+int(c)                 donne 50       ( 3+47)
```

E1

Racine carrée de x	**sqrt(x)**
Les fonctions trigonométriques	**cos(x), sin(x),tan(x),acos(x),asin(x), ...**
Partie entière de x	**floor(x)**
Valeur absolue de x	**abs(x)**
Valeur de π	**pi**

```
a= input("Longueur AC ?")
b=input("Longueur AB ?")
h=sqrt(a**2+b**2)
print "L'hypotenuse mesure",h
print"L'angle B mesure ",floor(acos(b/h)),'°'
print"L'angle C mesure ",floor(asin(b/h)),'°'


print"Perimetre du cercle circonscrit ",pi*h
print"Aire du cercle circonscrit ",pi**2*(h/2
```

```
>>>
L'hypotenuse mesure 5.0
L'angle B mesure  36.0 °
L'angle C mesure  53.0 °
Perimetre du cercle circonscrit  15.7079632679
Aire du cercle circonscrit  24.6740110027
>>>
```

E2

```
for i in range (1,21):
    print 'lancer',i,':',randint(1,6)
```

E3 1,2,3

```
import random

# inititialisations
prix = random.randint(1,100)
proposition = 0
coup = 1


while proposition != prix :
    print "Tentative n°",coup
          # Permet a la fois un affichage  plus convivial
          # et une verification du nombre de boucles

    proposition = input("Entrez un prix")
    if proposition < prix :
        print "C'est plus que",proposition

    elif proposition > prix :
        print "C'est moins que",proposition
    else :
        print "Vous avez gagné en ",coup,"coup(s)"

    if coup == 10 :
        print "Trop tard, la réponse était",prix
        proposition = prix
    coup = coup +1
```