ACADEMIE DE VERSAILLES
LYCEE DE L'ESSOURIAU

TD / TP Informatique (langage Python)

Algorithmique, tests et boucles

Tâche préliminaire : créer un sous-dossier de travail nommé « TP2 », dans le dossier à votre nom (par exemple « DUPONT_Pierre\Informatique\TP2 »).

Ce dossier devra regrouper l'ensemble des exercices réalisés pendant la séance de TP. Le code python correspondant à chaque exercice devra être placé dans un fichier source indépendant (nommé par exemple « exerciceXX.py »).

Vous utiliserez l'environnement de développement *Spyder* pour éditer, tester et debugger vos programmes.

Exercice 1 : Racines d'un polynôme du 2nd degré

- Saisir le programme de calcul des racines d'un polynôme vu en cours.
- Vérifier son fonctionnement.
- Modifier le programme afin qu'il calcule et affiche les racines complexes conjuguées du polynôme le cas échéant. On proposera des variables spécifiques permettant de mémoriser les valeurs complexes.

Exercice 2 : Sélection d'une valeur

On souhaite réaliser un programme qui affiche « Rouge », « Vert » ou « Bleu » si l'utilisateur saisit respectivement les caractères 'r' ou 'R', 'v' ou 'V', 'b' ou 'B'. Si un autre caractère est saisi, le programme affiche le message « Couleur non reconnue ».

- Proposer et tester un code répondant au cahier des charges ci-dessus.

Exercice 3 : Algorithme inconnu

On donne l'algorithme suivant, travaillant avec deux valeurs a et b en entrée (*a* et *b* étant des entiers $\in \mathbb{N}$) :

```
Variables a, b, m, p : entiers positifs ou nuls
Début
p ← 0
m ← 0
Tant que m < a, faire
    p ← p + b
    m ← m + 1
Fin Tant que
Fin
```

- **Sans utiliser l'ordinateur**, analyser l'algorithme ci-dessus et donner son rôle. On pourra pour cela « simuler » intellectuellement son fonctionnement pour plusieurs cas concrets (*a* et *b* sont alors à choisir).
- Vérifier l'affirmation en écrivant le code Python correspondant à l'algorithme et en l'exécutant.

Exercice 4 : Elévation d'un nombre à la puissance n

On souhaite réaliser un programme qui calcule a^n, n étant un entier positif ou nul. Ce programme **ne doit pas utiliser** l'opérateur ** !

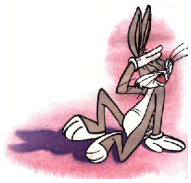
- Proposer un algorithme (écrit en pseudo-code) qui réponde à la demande ci-dessus.
- Prouver la terminaison et la validité de cet algorithme.
- Ecrire le code python correspondant et tester le fonctionnement du programme.

Exercice 5 : Comptage du nombre de chiffre composant un nombre entier

Soit un entier *nb* (positif), on souhaite réaliser un programme qui permet de déterminer le nombre de chiffres qui le composent.

Exemple : 6824 renvoie 4, 100 retourne 3, 8 retourne 1.

- Proposer un algorithme (écrit en pseudo-code) qui réponde à la demande ci-dessus.
- Ecrire le code python correspondant et tester le fonctionnement du programme.
- Amélioration : modifier le code précédent afin d'afficher successivement chacun des chiffres (unités, dizaines, centaines…)



Exercice 6 : Bugs Bunny et sa descendance…

Supposons qu'un couple (mâle-femelle) de lapins immatures soit mis dans un champ, que la maturité sexuelle du lapin soit atteinte après un mois qui est aussi la durée de gestation, que chaque portée comporte toujours un mâle et une femelle et que les lapins ne meurent pas.

Le problème est de savoir combien il y aura de lapins dans le champ après *n* années?

Soit u_n le nombre de couples de lapins au n-ième mois. On pose $u_0=1$et $u_1=1$ et on peut montrer que $u_n=u_{n-2} + u_{n-1}$ (suite de Fibonacci).

- Proposer un algorithme (écrit en pseudo-code) qui permette de calculer le terme u_n pour un rang *n* quelconque.
- Ecrire le code python correspondant et tester le fonctionnement du programme.
- Utiliser le programme afin de déterminer combien on a de lapins au bout de 2 ans.
- Et au bout de 4 ans ?

Exercice 7 : Spécial enseignant… Moyenne de classe (optionnel)

Réaliser un programme qui :

- Demande à l'utilisateur de choisir un nombre de notes à saisir
- Demande successivement de rentrer ces notes
- Calcule la moyenne
- L'affiche à l'écran

Remarque : on n'oubliera pas de vérifier que les valeurs saisies sont bien comprises entre 0 et 20.

Exercice 8 : Nombre inconnu…

On souhaite concevoir un jeu dont le principe est le suivant : l'utilisateur doit trouver un nombre choisi aléatoirement entre 0 et 100 par l'ordinateur, en moins de 10 essais. Le programme indique, pour chaque proposition, si le nombre à trouver est inférieur ou supérieur à la valeur proposée.

- Proposer un algorithme (écrit en pseudo-code) qui réponde à la demande ci-dessus.
- Ecrire le code python correspondant et tester le fonctionnement du programme.

Remarque : la fonction *randint(min, max)*, appartenant à la bibliothèque *random* (à importer…) , retourne un entier pris au hasard entre les valeurs *min* et *max* incluses.

Exercice 9 : Jeu de dés

On souhaite créer un jeu de dés sur ordinateur. Le jeu se joue à deux joueurs, le gagnant étant celui qui le premier atteint 50 points.

Un tour se déroule de la façon suivante :
Un joueur lance le dé et rejoue tant qu'il n'obtient pas de '1'.
A chaque lancé :

- S'il obtient un chiffre pair (2, 4, 6), il augmente ses points du chiffre obtenu
- S'il obtient un '3', ses points sont multipliés par 2
- S'il obtient un '5', il perd deux points.

Il perd la main quand il obtient un '1 '.

- Ecrire un organigramme pour le tour d'un joueur. On précisera les variables nécessaires et leur type.
- Ecrire le programme correspondant et le tester.
- Ecrire un organigramme représentant une partie complète. On utilisera l'organigramme précédent sous la forme d'un sous-organigramme. On précisera les variables nécessaires et leur type.
- Ecrire le programme complet correspondant.

Remarque : la fonction *randint(min, max)*, appartenant à la bibliothèque *random* (à importer…) , retourne un entier pris au hasard entre les valeurs *min* et *max* incluses. Elle pourra être utilisée pour simuler le lancer de dé.