TP2 Algorithmique/Python : utilisation de tests…

Exemple de programme en Python utilisant un test :

① `temp=float(input("Quelle température fait-il ?"))`
② `if temp<=0 :`
③ `print("L'eau gèle")`
`print("Attention au verglas")`
④ `else :`
`print("Aucun risque de verglas")`
⑤ `print("Fin du contrôle de température")`
	① La machine affiche le texte "Quelle température fait-il ?" et demande à l'utilisateur d'entrer une réponse. Cette réponse est un *texte*, qui est converti en *nombre* par float(…).
	Enfin, la réponse est stockée dans la variable *temp*.
	
	② Python regarde si (if) la valeur dans temp est inférieure ou égale à 0.
	
	③ Si oui, Python affiche "L'eau gèle" et "Attention au verglas". Les deux instructions sont décalées avec une tabulation pour indiquer qu'elle doivent être exécutées si temp ≤ 0.
	
	④ Si non (else:) Python affiche "Aucun risque de verglas". Là encore l'instruction est décalée.
	
	⑤ Dans tous les cas (pas de décalage), Python affiche "Fin du contrôle de température".

Remarque :
En Python, pour écrire une condition (qui suit le if), on utilise un des comparateurs suivants :

==			pour tester l'égalité
!=			pour tester une non égalité
<	ou	>	pour tester une infériorité ou supériorité stricte
<=	ou	>=	pour tester une infériorité ou supériorité large
in			pour tester une appartenance à un ensemble

Exercice 1 : compréhension d'algorithmes

1°) Voici deux algorithmes écrits en langage naturel (ou pseudo-code) et un écrit en Python. Indiquez la valeur de la variable S à la fin de chacun d'entre eux.

```
5 → B
– 3 → S
Si  B < S
Alors
        2 → S
Sinon
        5 → S
FinSi
```

```
1 → S
5 → D
– 3 → F
Si  F > S – D
Alors
        2 + S + F → S
Sinon
        5 + S + D → S
FinSi
```

```
E = 4
F = 2
if F!= E:
        F = F – E + 1
        S = E + F + 5
else:
        S = 2 + E + F
```

2°) Voici un algorithme :

```
Demander A
Demander B
Si A > B
Alors  Afficher A
Sinon
        Si A < B
        Alors   Afficher B
        Sinon
                Afficher « C'est bon »
        FinSi
FinSi
```

a) Qu'affiche cet algorithme si on entre les valeurs 1 et 4 ?
b) Même question si on entre les valeurs 5 et 2.
c) Dans quel cas l'algorithme affichera-t-il « C'est bon » ?

Exercice 2 : quelques petits programmes en Python

Programme 1 : Le code pin du téléphone de Marc est 4567.
Imaginez un programme exécuté par ce téléphone et qui demande à Marc son code pin puis vérifie s'il est correct (dans ce cas, le téléphone affichera juste « code correct », sinon il affichera « code incorrect »).

Programme 2 : La machine demande à un utilisateur son âge et dit si l'utilisateur est un enfant (moins de 12 ans) ou un adolescent (entre 12 et 18 ans) ou un adulte (à partir de 18 ans).

Exercice 3 : en géométrie

Écrivez un programme qui demande les longueurs des trois côtés d'un triangle et dit si ce triangle est rectangle ou pas.

Le hasard avec Python

Il est souvent utile de créer des nombres aléatoires, que ce soit pour des simulations informatiques ou pour des jeux.
Python ne sait pas faire cela à la base, on doit donc ajouter à Python de nouvelles fonctions.

Il y a pour cela deux instructions possibles, à taper au début du programme :
soit :

```
>>> from random import random
```

qui veut dire : dans la bibliothèque random, aller chercher la fonction random.
soit :

```
>>> import random
```

qui veut dire : aller chercher toutes les fonctions de la bibliothèque random.

Vous pouvez ensuite utiliser la fonction random() ainsi :

```
>>> random.random()
```

(utilisez la flèche vers le haut du clavier pour relancer cette commande)

Si vous avez importé la bibliothèque complète, vous aurez aussi accès à la fonction randint(), qui choisit un nombre entier au hasard, par exemple pour simuler un lancer de dé, tapez :

```
>>> random.randint(1,6)
```

Remarque : nous avions déjà vu ce mécanisme d'importation dans le TP1 avec la fonction sqrt() de la bibliothèque math, qu'on importe avec from math import sqrt puis qu'on utilise en tapant math.sqrt(…).

Exercice 4 : un petit jeu

Écrivez un jeu qui procède ainsi :

- la machine choisit un nombre mystère entre 1 et 100 ;
- un premier joueur entre un nombre dans la machine ;
- un second joueur entre un autre nombre dans la machine ;
- la machine dit lequel des deux a été le plus proche du nombre mystère.

Une opération utile

Tapez dans la console les instructions suivantes :

```
>>> 6 % 3
>>> 7 % 3
>>> 8 % 3
>>> 9 % 3
```

L'opérateur % donne le reste de la division d'un nombre par un autre.
Par exemple, $6 = 2 \times 3 + 0$; $7 = 2 \times 3 + 1$; $8 = 2 \times 3 + 2$; etc.
% est très utile pour savoir si un nombre est divisible par un autre.

Par exemple :

```
>>> a % 2
```

donnera 0 si a est pair et donnera 1 si a est impair.

Exercice 5 : les années bissextiles

Vous le savez sans doute, une année fait environ 365,25 jours.
La plupart des années calendaires font 365 jours, tandis que certaines (dites bissextiles) en font 366.

L'année sera bissextile :

1. si l'année est divisible par 4 et non divisible par 100,
   ou
2. si l'année est divisible par 400.

Sinon, l'année n'est pas bissextile (elle a 365 jours).

Écrivez un programme qui demande à un utilisateur de saisir une année puis qui dit si celle-ci est bissextile ou pas.

And et Or

On doit parfois écrire des conditions plus compliquées, utilisant les mots-clés and et or (et et ou).

L'exemple précédent pouvait ainsi s'écrire :

```
Si (annee est divisible par 400) ou (si annee est divisible par 4 et annee non divisible par 100)
Alors
        Afficher « Année bissextile »
Sinon
        Afficher « Année non bissextile »
FinSi
```

Exercice 5bis : les années bissextiles

Ré-écrivez votre programme de l'exercice 5 en utilisant and et or.

Exercice 6 : voyage scolaire

Un lycée organise un voyage scolaire en Espagne.
Seuls les élèves suivants peuvent y participer :

- tous les élèves de seconde qui pratiquent l'espagnol ;
- tous les élèves de première désireux de voyager.

Écrivez un programme qui demande à un élève quelle est sa classe, puis qui lui demande s'il pratique l'espagnol puis lui dit s'il peut participer au voyage.

Exercice bonus : le jour de votre naissance

Voici une méthode pour déterminer le jour de la semaine d'une date donnée entre 1900 et 2099 :

– Déterminer le code du mois avec le tableau de correspondance suivant :

Janvier	Février	Mars	Avril	Mai	Juin	Juillet	Août	Septembre	Octobre	Novembre	Décembre
0	3	3	6	1	4	6	2	5	0	3	5

– Ajouter le nombre formé par les « deux chiffres de droite » de l'année, le quart de ce nombre (tronqué à l'entier inférieur), le numéro du jour et le code du mois.
– Si la date est après 2000, enlever 1 au résultat.
– Si l'année est bissextile et si la date est avant le 1er mars, ajouter 1 au résultat.
– Calculer le reste de la division du résultat obtenu par 7.
– Convertir ce reste en jour avec la correspondance : 0 → dimanche ; 1 → lundi ; 2 → mardi…

Cette méthode permet par exemple de savoir quel jour était le 6 juin 1944 (année bissextile).
Le code du mois est 4, donc le calcul est : 44 + 11 + 6 + 4 = 65.
Le reste de la division de 65 par 7 étant 2, ce jour était un mardi.

Écrivez un programme qui demande à un utilisateur sa date de naissance et lui dit quel jour de la semaine il est né.

Il y a trois sortes de tests :

- le test « Si … alors » s'écrit ainsi :

```
if condition :
        instruction1
        instruction2
        etc.
suite du programme
```

- le test « Si … alors … sinon … » s'écrit ainsi :

```
if condition :
        instruction1
        instruction2
        etc.
else :
        instruction_alternative1
        instruction_alternative2
        etc.
suite du programme
```

- le test « Si … alors … sinon si … sinon … » s'écrit ainsi :

```
if condition :
        instruction1
        instruction2
        etc.
elif autre_condition :
        instruction_alternative1
        instruction_alternative2
        etc.
else :
        instruction_alternative1bis
        instruction_alternative2bis
        etc.
suite du programme
```

On utilise les tabulations pour décaler les instructions qui doivent être exécutées en cas de réalisation de la condition. On dit que ces instructions sont indentées.

Les conditions s'écrivent : *variable* *symbole* *variable ou nombre ou texte ou calcul*

où *symbole* est un des symboles suivants :

< > <= >= == != in