Algorithmes et structures de données : TD 6(sur machine)

Châines de caractères - Entrées de l'utilisateur - Boucles consécutives et imbriquées

Notez :
`a = input()` affecte la variable `a` avec une variable de type entier, flottant, ou booléen.
`a = raw_input()` affecte la variable `a` avec une variable de type chaîne de caractères.
Vous pouvez trouver les cours et TDs sur http://www.labri.fr/∼preuter/asd2009

Exercice 6.1 *Entrée d'une châine de caractère du clavier*

1. Ecrire un programme en python qui demande à l'utilisateur son prénom, et qui affiche un texte de bienvenu personnalisé (par exemple, Bonjour Anna).
2. Afficher à l'écran le nombre de caractères dans le prénom renseigné.
3. Les prénoms qui terminent par un 'a' sont souvent de prénoms féminins. Pour ces noms renseignés, afficher également le message "Vous êtes probablement féminin".
4. Rajouter une régle equivalente pour des prénoms féminins qui se terminent par 'ie'.

```
print "Entrez votre prénom : ",
prenom = raw_input();
print "Bonjour ",prenom
print
nombre = len(prenom)
print "Votre prénom a ", nombre, " caractères."

if (len(prenom)>=1):
    derniereLettre = prenom[len(prenom)-1]
    if (derniereLettre == "a"):
        print "Vous êtes probablement féminin"


if (len(prenom)>=2):
    derniereLettre = prenom[len(prenom)-1]
    avantDerniereLettre = prenom[len(prenom)-2]
    if (avantDerniereLettre=="i" and derniereLettre == "e"):
        print "Vous êtes probablement féminin"
```

Exercice 6.2 *Entrée numériques de l'utilisateur*

1. Ecrire un programme en python qui demande à l'utilisateur "Saisir un nombre", et qui affecte la variable `x` avec le nombre saisi. Le programme calculera ensuite le carré du nombre renseigné et affichera la réponse à l'écran. Exemple :

```
Le carré de 11 est égale à 121.
```

```
print "Entrez un nombre : ",
nombre = input()
carre = nombre*nombre
print "Le carré de",nombre,"est égale à",carre
```

Exercice 6.3 *Boucles*

1. Ecrire un algorithme qui affiche à l'écran le pattern suivant. Utiliser deux boucles successives !

```
Bonjour
Bonjour
Bonjour
Bonjour
Salut
Salut
Salut
```

2. Modifier votre algorithme pour qu'il demande à l'utilisateur de saisir un nombre entier et qui le stocke dans la variable `b`, et qu'il demande à l'utilisateur de saisir un autre nombre entier et qui le stocke dans la variable `s`. L'algorithme devrait ensuite afficher `b` fois "Bonjour" et `s` fois "Salut".

```
print "Entrez un nombre pour bonjour : ",
b = input()
print "Entrez un nombre pour salut : ",
s = input()
i=1
while i<=b:
    print "Bonjour"
    i=i+1
i=1
while i<=s:
    print "Salut"
    i=i+1
```

Exercice 6.4 *Chaînes de caractères* ***en python***

Dans beaucoup de programmes, les utilisateurs doivent choisir un mot de passe. Pour la sécurité de l'utilisateur, les programmes imposent de choisir un mot de passe qui n'est pas trop court.

1. Ecrire un programme **en python** qui demande à l'utilisateur "Entrer votre mot de passe", et qui permet ensuite à l'utilsateur de saisir un mot de passe. Si le mot de passe contient moins de 6 caractères, afficher à l'écran le texte "Le mot de passe est trop court". Sinon, afficher à l'écran le texte "Le mot de passe choisi est accepté".

```
print "Entrer votre mot de passe : "
mdp = raw_input()
longueur = len(mdp)
if (longueur >= 6):
    print "Le mot de passe est accepté"
else:
    print "Le mot de passe est trop court"
```

2. Augmenter la sécurité en imposant que le mot de passe contienne au moins un de ces trois caractères spéciaux : ('_', '-', ou ':').

```
print "Entrer votre mot de passe : "
mdp = raw_input()
longueur = len(mdp)
if (longueur >= 6):
    i = 0
    accepte = False
    while (i<longueur):
        if (mdp[i] == "_" or mdp[i] == "-" or mdp[i] == ":"):
            accepte =  True
        i = i + 1
    if (accepte == True):
        print "Le mot de passe est accepté"
    else:
        print "Le mot de passe doit contenir un caractère  spécial _, ou -, ou ;"
else:
    print "Le mot de passe est trop court"
```

3. Améliorer le programme pour qu'il demande à l'utilisateur d'entrer un mot de passe, jusqu'à ce qu'il soit valide.

```
accepte = False
while (accepte == False):
    print "Entrer votre mot de passe : "
    mdp = raw_input()
    longueur = len(mdp)
    if (longueur >= 6):
        i = 0
        while (i<longueur):
            if (mdp[i] == "_" or mdp[i] == "-" or mdp[i] == ":"):
                accepte =  True
            i = i + 1
        if (accepte == True):
            print "Le mot de passe est accepté"
        else:
            print "Le mot de passe doit contenir un caractère  spécial _, ou -, ou ;"

    else:
        print "Le mot de passe est trop court"
```

Exercice 6.5 *Boucles imbriquées*

1. Considérer l'algorithme suivant :

```
n = 5
i = 1
TANT QUE i<=n FAIRE
    j = 1
    TANT QUE j<=n FAIRE
        SI (j<=i) ALORS
            afficher "1",
        SINON
            afficher "0",
        FIN SI
        j = j + 1
    FIN TANT QUE
    afficher
    i = i + 1
FIN TANT QUE
```

2. Ecrire le programme en python qui correspond à cet algorithme.

```
n = 5
i = 1
while i<=n:
    j = 1
    while j<=n:
        if j<=i:
            print "1",
        else:
            print "0",
        j = j + 1
    print
    i = i + 1
```

3. Modifier l'algorithme pour quelques valeurs pour `n`.

Exercice 6.6 *Convertisseur*

1. Ecrire un algorithme qui convertit un nombre entier de secondes fourni au départ dans la variable `secondes`, en nombre de minutes, et de secondes.
2. Tester votre algortithme pour l'affectation suivante :

```
secondes = 1331
```

```
secondes = 1331

min = secondes / 60
sec = secondes % 60
#ou bien
#sec = secondes - (min * 60)
print secondes," equivaut ", min, " minutes et ",sec, " secondes"
```

3. Changer votre algortithme pour qu'il convertisse le nombre entier de secondes fourni au départ dans la variable `secondes`, en nombre d'heures, de minutes, et de secondes.
4. Tester votre algortithme pour l'affectation suivante :

```
secondes = 18243
```

```
secondes = 18243

heures = secondes / 3600
sec = secondes % 3600

min = sec / 60
sec = sec % 60
print secondes," equivaut ", heures, " heures et ",min, " minutes et ",sec, " secondes"
```

5. Changer votre algortithme pour qu'il convertisse le nombre entier de secondes fourni au départ dans la variable `secondes`, en nombre de jours, d'heures, de minutes, et de secondes.
6. Tester votre algortithme pour l'affectation suivante :

```
secondes = 448402
```

```
secondes = 448402

jours = secondes / (3600*24)
sec = secondes % (3600*24)

heures = sec / 3600
sec = sec % 3600

min = sec / 60
sec = sec % 60

print secondes," equivaut ",jours, " jours et ",heures, " heures et ",min, " minutes et ",sec, " secondes"
```

7. Modifier votre algorithme pour que l'utilisateur puisse entrer la valeur pour `secondes` en début de programme.

```
secondes = input()

jours = secondes / (3600*24)
sec = secondes % (3600*24)

heures = sec / 3600
sec = sec % 3600

min = sec / 60
sec = sec % 60

print secondes," equivaut ",jours, " jours et ",heures, " heures et ",min, " minutes et ",sec, " secondes"
```