

Cours 4 : Python, listes et boucles

The image shows three overlapping screenshots of a Python Shell and IDLE 1.1 interface. The top-left window shows the Python Shell output for a Python 2.4 installation. The middle window shows the same output with a firewall warning. The bottom-right window shows the IDLE 1.1 prompt with several Python code snippets and their outputs.

```

Python 2.4 (#2, Feb 12 2005, 00:29:46)
[GCC 3.4.3 (Mandrakelinux 10.2 3.4.3-3mdk)] on linux2
Type "copyright", "credits" or "license()" for more information.

*****
Personal firewall software may warn about the connection IDLE
makes to its subprocess using this computer's internal loopback
interface. This connection is not visible on any external
interface and no data is sent to or received from the Internet.
*****

IDLE 1.1
>>> print 1 + 2
3
>>> print 6 - 1
5
>>> niveau_de_risque = 2
>>> risque_due_au_tabac = 4
>>> niveau_de_risque = niveau_de_risque + risque_due_au_tabac
>>> print niveau_de_risque
6
>>> |

Python 2.4 (#2, Feb 12 2005, 00:29:46)
[GCC 3.4.3 (Mandrakelinux 10.2 3.4.3-3mdk)] on linux2
Type "copyright", "credits" or "license()" for more information.

*****
Personal firewall software may warn about the connection IDLE
makes to its subprocess using this computer's internal loopback
interface. This connection is not visible on any external
interface and no data is sent to or received from the Internet.
*****

IDLE 1.1
>>> print 1 + 2
3
>>> print 6 - 1
5
>>> niveau_de_risque = 2
>>> risque_due_au_tabac = 4
>>> niveau_de_risque = niveau_de_risque + risque_due_au_tabac
>>> print niveau_de_risque
6
>>> |

Python 2.4 (#2, Feb 12 2005, 00:29:46)
[GCC 3.4.3 (Mandrakelinux 10.2 3.4.3-3mdk)] on linux2
Type "copyright", "credits" or "license()" for more information.

*****
Personal firewall software may warn about the connection IDLE
makes to its subprocess using this computer's internal loopback
interface. This connection is not visible on any external
interface and no data is sent to or received from the Internet.
*****

IDLE 1.1
>>> print 1 + 2
3
>>> print 6 - 1
5
>>> niveau_de_risque = 2
>>> risque_due_au_tabac = 4
>>> niveau_de_risque = niveau_de_risque + risque_due_au_tabac
>>> print niveau_de_risque
6
>>> |

```

Ln: 22 Col: 4

Listes

- Les listes contiennent un tableau de données :
`patients = ["Jean Dupont", "Paul Dubois", "Marianne Y"]`
- Il est possible de mélanger des types différents :
`chromosomes_recombines = [1, 3, 5, 12, 25, "X"]`
- Opérations :
`print len(chromosomes_recombines) -> 6`
`print [0, 1, 5] + [2, 3] -> [0, 1, 5, 2, 3]`
`print chromosomes_recombines[0] -> 1`
`chromosomes_recombines.append(14)`
`chromosomes_recombines.remove("X")`
`chromosomes_recombines.sort()`

`if "Y" in chromosomes:`
 `print "C'est un garçon !"`

Listes

- Attention, «=» ne copie pas les listes !

```
patients = ["Jean Dupont", "Paul Dubois", "Marianne Y"]  
patients2 = patients  
patients2.remove("Jean Dupont")
```

```
print patient2
```

```
-> ["Paul Dubois", "Marianne Y"]
```

```
print patient
```

```
-> ["Paul Dubois", "Marianne Y"]
```

- Pour copier une liste :

```
patients3 = patients[:]
```

Listes

- Exercice
 - Créer une liste d'ADN vide
`adns = []`
 - Ajouter dans cette liste les ADN "atcgta", "cct" et "agc"

 - Afficher le nombre d'ADN dans cette liste

Listes

- Exercice

- Créer une liste d'ADN vide

```
adns = [ ]
```

- Ajouter dans cette liste les ADN "atcgta", "cct" et "agc"

```
adns.append("atcgta")
```

```
adns.append("cct")
```

```
adns.append("agc")
```

- Afficher le nombre d'ADN dans cette liste

```
print len(adns)
```

Tuples

- Un tuple est similaire à une liste, mais ne peut pas être modifié
- Il s'écrit avec des parenthèses () au lieu des crochets []
`patients = ("Jean Dupont", "Paul Dubois", "Marianne Y")`

Boucles

- Les boucles permettent d'exécuter plusieurs fois les mêmes ordres

- Pour parcourir une liste :

```
for variable in liste:
```

```
    code de la boucle
```

- Exemple :

```
for chromosome in chromosomes_anormaux:
```

```
    print "le chromosome", chromosome, "est anormal !"
```

```
-> le chromosome 1 est anormal !
```

```
-> le chromosome 3 est anormal !
```

```
-> ...
```

```
for base in adn:
```

```
    print base
```

```
-> a...
```

Boucles

- `range([debut], fin, [pas])` permet d'obtenir une liste de nombre

```
print range(10)           -> [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

```
print range(4, 10)       -> [4, 5, 6, 7, 8, 9]
```

```
print range(4, 10, 2)    -> [4, 6, 8]
```

- Exemple :

```
for i in range(10):
```

```
    print i
```

```
-> 0
```

```
-> 1
```

```
-> 2
```

```
-> ..
```

```
-> 9
```

Boucles

- Range peut être utilisé pour boucler sur les indices, et non pas sur les éléments d'une liste

- Exemple :

```
adn = "atcacgtta"
```

```
for i in range(len(adn)):
```

```
    print "la base n°", i, "est", adn[i]
```

```
-> la base n° 0 est a
```

```
-> la base n° 1 est t
```

```
-> la base n° 2 est c
```

```
-> ...
```

```
-> la base n° 8 est a
```

Boucles

- Pour boucler tant qu'une condition est vérifiée :
`while condition:`
`code de la boucle`
- `1` est une condition toujours vraie, `break` quitte la boucle
- Exemple :

```
maladies = []  
while 1:  
    nouvelle_maladie = raw_input("Entrez la maladie : ")  
    if nouvelle_maladie == "":  
        break  
    maladies.append(nouvelle_maladie)  
print maladies
```

Boucles

- Exercice
 - Dans la séquence protéique suivante (utilisant le code international des acides aminés), compter le nombre de Cystéines (code C)

```
proteine = "CVAPGPMCAWCDSTAC"
```

Boucles

- Exercice
 - Dans la séquence protéique suivante (utilisant le code international des acides aminés), compter le nombre de Cystéines (code C)

```
proteine = "CVAPGPMCAWCDSTAC"
```

```
nb_cysteine = 0
```

```
for aa in proteine:
```

```
    if aa == "C":
```

```
        nb_cysteine = nb_cysteine + 1
```

```
print "il y a", nb_cysteine, "Cystéine"
```

Boucles

- Exercice
 - Dans la séquence protéique suivante (utilisant le code international des acides aminés), compter le nombre d'atomes de soufre

```
proteine = "CVAPGPMCAWCDSTAC"
```

Boucles

- Exercice

- Dans la séquence protéique suivante (utilisant le code international des acides aminés), compter le nombre d'atomes de soufre
- rappel : les acides aminés soufrés sont la Cystéine (C) et la Méthionine (M)

```
proteine = "CVAPGPMCAWCDSTAC"
```

```
nb_soufre = 0
```

```
for aa in proteine:
```

```
    if (aa == "C") or (aa == "M"):
```

```
        nb_soufre = nb_soufre + 1
```

```
print "il y a", nb_soufre, "atomes de soufre"
```

Boucles

- Exercice

- Sur les protéines, les angles dièdres phi/psi d'une hélice alpha parfaite ont une valeur de -57° et -47° respectivement, $\pm 30^\circ$.
- La liste de listes suivante contient les valeurs de phi/psi de la première hélice de la chaîne 1tfe. Écrire un programme qui teste pour chacun des résidus s'ils sont en hélice ou non.

```
angles_diedres = [[48.6, 53.4], [-124.9, 156.7], [-66.2, -30.8], [-58.8, -43.1], [-73.9, -40.6], [-53.7, -37.5], [-80.6, -16.0], [-68.5, 135.0], [-64.9, -23.5], [-66.9, -45.5], [-69.6, -41.0], [-62.7, -37.5], [-68.2, -38.3], [-61.2, -49.1], [-59.7, -41.1], [-63.2, -48.5], [-65.5, -38.5], [-64.1, -40.7], [-63.6, -40.8], [-66.4, -44.5], [-56.0, -52.5], [-55.4, -44.6], [-58.6, -44.0], [-77.5, -39.1], [-91.7, -11.9], [48.6, 53.4]]
```

Boucles

- Exercice

- Sur les protéines, les angles dièdres phi/psi d'une hélice alpha parfaite ont une valeur de -57° et -47° respectivement, $\pm 30^\circ$.
- La liste de listes suivante contient les valeurs de phi/psi de la première hélice de la chaîne 1tfe. Écrire un programme qui teste pour chacun des résidus s'ils sont en hélice ou non.

```
for angle in angles_diedres:
```

```
    phi = angle[0]
```

```
    psi = angle[1]
```

```
    if (-57.0 - 30.0 < phi < -57.0 + 30.0) and (-47.0 - 30.0 < psi < -47.0 + 30.0):
```

```
        print "le résidu est en hélice alpha"
```

```
    else:
```

```
        print "le résidu n'est pas en hélice alpha"
```

Boucles

- Exercice
 - Demander à l'utilisateur d'entrer une chaîne d'ADN et vérifier que cette chaîne est bien une chaîne d'ADN, c'est-à-dire qu'elle contient seulement des "A", des "T", des "C" et des "G"

Boucles

- Exercice

- Demander à l'utilisateur d'entrer une chaîne d'ADN et vérifier que cette chaîne est bien une chaîne d'ADN, c'est-à-dire qu'elle contient seulement des "A", des "T", des "C" et des "G"

```
adn = raw_input("Entrez la chaîne : ")
```

```
for base in adn:
```

```
    if (base != "A") and (base != "T") and (base != "C") and (base != "G"):  
        print "Ce n'est pas un ADN !"  
        break
```

Boucles

- Exercice

- Demander à l'utilisateur d'entrer une chaîne d'ADN et vérifier que cette chaîne est bien une chaîne d'ADN, c'est-à-dire qu'elle contient seulement des "A", des "T", des "C" et des "G"

```
adn = raw_input("Entrez la chaîne : ")
```

```
est_un_adn = 1
```

```
for base in adn:
```

```
    if (base != "A") and (base != "T") and (base != "C") and (base != "G"):
```

```
        print "Ce n'est pas un ADN !"
```

```
        est_un_adn = 0
```

```
        break
```

```
if est_un_adn == 1:
```

```
    print "C'est un ADN !"
```

Boucles

- Exercice
 - Demander à l'utilisateur d'entrer une chaîne d'ADN et vérifier que cette chaîne est bien une chaîne d'ADN, c'est-à-dire qu'elle contient seulement des "A", des "T", des "C" et des "G"

```
adn = raw_input("Entrez la chaîne : ")
```

```
for base in adn:
```

```
    if (base != "A") and (base != "T") and (base != "C") and (base != "G"):  
        print "Ce n'est pas un ADN !"  
        break
```

```
else:
```

```
    print "C'est un ADN !"
```

Boucles

- Exercice
 - Demander à l'utilisateur d'entrer une liste de numéros de chromosome, puis indiquer si la personne ayant ces chromosomes est un homme ou une femme.
 - Comment faire pour demander à l'utilisateur d'entrer une liste ?

Boucles

- Exercice
 - Demander à l'utilisateur d'entrer une liste de numéros de chromosome, puis indiquer si la personne ayant ces chromosomes est un homme ou une femme.
 - Comment faire pour demander à l'utilisateur d'entrer une liste ?
 - Créer une liste vide
 - L'utilisateur entre un premier chromosome
 - Tant que la valeur entrée est valide, on l'ajoute dans la liste et on demande d'entrer un nouveau chromosome
 - Si la valeur entrée est vide, la liste est terminée et on passe à la suite

Boucles

- Exercice
 - Demander à l'utilisateur d'entrer une liste de numéros de chromosome, puis indiquer si la personne ayant ces chromosomes est un homme ou une femme.

```
chromosomes = []  
while 1:  
    nouveau_chromosome = raw_input("Entrez un n° de chromosome : ")  
    if nouveau_chromosome == "": break  
    chromosomes.append(nouveau_chromosome)  
  
if "Y" in chromosomes: print "C'est un homme."  
else: print "C'est une femme."
```

Boucles

- Exercice
 - Demander à l'utilisateur d'entrer une liste de numéros de chromosome, puis indiquer si la personne ayant ces chromosomes est atteinte de trisomie 21.

Boucles

- Exercice
 - Demander à l'utilisateur d'entrer une liste de numéros de chromosome, puis indiquer si la personne ayant ces chromosomes est atteinte de trisomie 21.

```
chromosomes = []
while 1:
    nouveau_chromosome = raw_input("Entrez un n° de chromosome : ")
    if nouveau_chromosome == "": break
    chromosomes.append(nouveau_chromosome)

nb_21 = 0
for chromosome in chromosomes:
    if chromosome == "21": nb_21 = nb_21 + 1
if nb_21 == 3:
    print "la personne est atteinte de trisomie 21"
```