

# Cours 6 : Dictionnaire

The image shows three overlapping windows of a Python Shell and IDLE 1.1. The top-left window shows the Python Shell output for a Python 2.4 installation. The middle window shows IDLE 1.1 code for arithmetic and variable assignment. The bottom-right window shows IDLE 1.1 code for arithmetic, variable assignment, and dictionary creation.

```

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
>>> |

```

# Dictionnaires

- Association clef-valeur :

```
definitions = { "os" : "système d'exploitation",  
               "python" : "langage de programmation",  
               "dictionnaire" : "association clef-valeur" }
```

- Opérations :

```
print len(definitions)           -> 3  
print definitions["python"]      -> langage de programmation  
print definitions.get("python","rien") -> langage de programmation  
print definitions.get("n'existepas","rien") -> rien  
print definitions.has_key("python") -> 1  
print definitions.has_key("n'existepas") -> 0  
definitions["flottant"] = "nombre décimal"  
del definitions["os"]
```

# Dictionnaires

- Exercice :
  - Créer un dictionnaire associant à chaque base de l'ADN (a, t, g, c) la chaîne "purinique" ou "pyrimidinique" (selon la base)
  - Demander à l'utilisateur d'entrer une base
  - Afficher si la base entrée par l'utilisateur est purinique ou pyrimidinique

# Dictionnaires

- Exercice :
  - Créer un dictionnaire associant à chaque base de l'ADN (a, t, g, c) la chaîne "purinique" ou "pyrimidinique" (selon la base)

```
types_de_base = {  
    "a" : "pyrimidinique",  
    "t" : "pyrimidinique",  
    "g" : "purinique",  
    "c" : "purinique",  
}
```

- Demander à l'utilisateur d'entrer une base

```
base = raw_input("Entrez une base : ")
```

- Afficher si la base entrée par l'utilisateur est purinique ou pyrimidinique

```
print "C'est une base", types_de_base[base]
```

# Dictionnaires

- Exercice : traduction d'un ARNm
  - Comment faire pour afficher la liste des codons présents dans une séquence d'ARNm ?

```
arn = "UCUAGGCUG"
```

# Dictionnaires

- Exercice : traduction d'un ARNm
  - Comment faire pour afficher la liste des codons présents dans une séquence d'ARNm ?

```
arn = "UCUAGGCUG"
for i in range(0, len(arn), 3):
    codon = arn[i : i + 3]
    print codon
```
  - Écrire une fonction permettant de traduire une chaîne d'ARNm en une chaîne protéique, en s'aidant du dictionnaire suivant :

```
code_genetique = { "UUU" : "F", "UUC" : "F", "UUA" : "L", "UUG" :
"L", "UCU" : "S", [...] }
```

# Dictionnaires

- Exercice : traduction d'un ARNm
  - Écrire une fonction permettant de traduire une chaîne d'ARNm en une chaîne protéique, en s'aidant du dictionnaire suivant :  
`code_genetique = { "UUU" : "F", "UUC" : "F", "UUA" : "L", "UUG" : "L", "UCU" : "S", [...] }`

```
def traduction(arn):  
    proteine = ""  
  
    for i in range(0, len(arn), 3):  
        codon = arn[i : i + 3]  
        acide_amine = code_genetique[codon]  
        proteine = proteine + acide_amine  
  
    return proteine
```

# Dictionnaires

- Exercice :
  - Écrire un programme pour compter le nombre de chacun des acides aminés dans une séquence protéique, et afficher le résultat

```
proteine = "MCVSLCLLSACCLLMVVLLC"
```

# Dictionnaires

- Exercice :
  - Écrire un programme pour compter le nombre de chacun des acides aminés dans une séquence protéique, et afficher le résultat

```
proteine = "MCVSLCLLSACCLLMVVLLC"
```

```
acides_amines = {}
```

```
for aa in proteine:
```

```
    acides_amines[aa] = acides_amines.get(aa, 0) + 1
```

```
for aa, nb in acides_amines.items():
```

```
    print nb, aa
```

# Dictionnaires

```
d = { "Escherichia coli" : 3.6, "Homo sapiens" : 3200,  
      "Saccharomyces cerevisiae" : 12, "Arabidopsis thaliana" : 125 }
```

- Pour récupérer la liste des clefs, des valeurs ou des couples (clef, valeur) :

```
print d.keys() -> ["Escherichia coli", "Homo sapiens",  
                  "Saccharomyces cerevisiae", "Arabidopsis thaliana"]
```

```
print d.values() -> [ 3.6, 3200, 12, 125 ]
```

```
print d.items() -> [ ("Escherichia coli", 3.6),  
                    ("Homo sapiens", 3200),  
                    ("Saccharomyces cerevisiae", 12),  
                    ("Arabidopsis thaliana", 125) ]
```

```
for key, value in d.items():
```

```
    print key, "a un génôme de", value, "paires de bases"
```

# Dictionnaires

- Exercice :
  - d est un dictionnaire faisant correspondre à chaque espèce la longueur de son génôme  
`d = { "Escherichia coli" : 3.6, "Homo sapiens" : 3200, "Saccharomyces cerevisiae" : 12, "Arabidopsis thaliana" : 125 }`
  - Rechercher dans ce dictionnaire l'organisme possédant le plus grand génôme et indiquer son nom

# Dictionnaires

- Exercice :
  - d est un dictionnaire faisant correspondre à chaque espèce la longueur de son génôme

```
d = { "Escherichia coli" : 3.6, "Homo sapiens" : 3200,  
      "Saccharomyces cerevisiae" : 12, "Arabidopsis thaliana" : 125 }
```

- Rechercher dans ce dictionnaire l'organisme possédant le plus grand génôme et indiquer son nom

```
plus_long = ""  
longueur_max = 0.0  
for espece, longueur in d.items():  
    if longueur > longueur_max:  
        longueur_max = longueur  
        plus_long = espece  
print plus_long
```