

Cours 5 : Fonctions

Python Shell

```

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 Shell

```

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 Shell

```

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

Fonctions

- Une fonction est un morceau de programme que l'on exécute ainsi :
`resultat = nom_de_la_fonction(paramètre1, paramètre2,...)`
- Nous avons déjà rencontré des fonctions :
`chaine_entree = raw_input(message)`
`entier = int(chaine)`
`flottant = float(chaine)`
`sequence_d_entiers = range(debut, fin, pas)`
- Certaines fonctions ne renvoient pas de résultat
- Avantage des fonctions : il est possible de les utiliser sans savoir comment elles marchent !

Fonctions

- Python contient un grand nombre de fonctions prédéfinies
- Exemple : fonction `help`
`help(raw_input)`
-> Help on built-in function raw_input in module `__builtin__`:
`raw_input([prompt])` -> string
Read a string from standard input. The trailing newline is stripped.
If the user hits EOF (Unix: Ctl-D, Windows: Ctl-Z+Return), raise EOFError. On Unix, GNU readline is used if enabled. The prompt string, if given, is printed without a trailing newline before reading.

Fonctions

- Il est aussi possible de définir ses propres fonctions :

```
def nom_de_la_fonction(paramètre1, paramètre2,...):  
    code de la fonction  
    return resultat
```

- Exemple :

```
def indice_de_masse_corporelle(poids, taille):  
    imc = poids / (taille * taille)  
    return imc
```

```
print indice_de_masse_corporelle(64.0, 1.70)
```

```
-> 22.145328719723185
```

- Les variables poids et tailles prennent les valeurs données :
64.0 et 1.70

Fonctions

- On peut aussi écrire :

```
def indice_de_masse_corporelle(poids, taille):  
    return poids / (taille * taille)
```

- Une fonction peut appeler d'autres fonctions :

```
def en_surpoids(poids, taille):  
    imc = indice_de_masse_corporelle(poids, taille)  
    if imc > 25.0:  
        return 1  
    else:  
        return 0
```

```
if en_surpoids(64.0, 1.70) == 1:  
    print "Il faut faire un régime !"
```

Fonctions

- La portée des variables définies dans une fonction est limitée à cette fonction

```
poids = 80.0
def indice_de_masse_corporelle(poids, taille):
    return poids / (taille * taille)
print indice_de_masse_corporelle(64.0, 170.0)
print poids
```

```
-> 22.145328719723185
```

```
-> 80.0
```

Fonctions

- Exercice :
 - Le pH d'une solution aqueuse d'acide faible est donné par la formule suivante :
$$\text{pH} = \frac{\text{pKa}}{2} - \frac{\log(\text{Ca})}{2}$$
 - Avec pKa le pKa de l'acide et Ca la concentration de cet acide en mol/l.
 - Écrire une fonction qui calcule le pH d'une solution aqueuse d'acide faible.
 - Pour calculer le logarithme de x : `math.log(x)`
 - Calculer le pH d'une solution d'acide acétique (pKa = 4,76) concentré à 0,1 mol/l.

Fonctions

- Exercice :
 - Le pH d'une solution aqueuse d'acide faible est donné par la formule suivante :
$$\text{pH} = \frac{\text{pKa}}{2} - \frac{\log(\text{Ca})}{2}$$
 - Écrire une fonction qui calcule le pH d'une solution aqueuse d'acide faible.
 - Calculer le pH d'une solution d'acide acétique (pKa = 4,76) concentré à 0,1 mol/l.

```
def calcule_pH_acide_faible(pKa, Ca):  
    pH = pKa / 2.0 - math.log(Ca) / 2.0  
    return pH
```

```
print calcule_pH_acide_faible(4.76, 0.1)
```

Fonctions

- Exercice :
 - Écrire une fonction appelée `est_adn`
 - qui prend comme paramètre une chaîne de caractère
 - qui retourne 1 si la chaîne est une chaîne d'ADN, et 0 sinon
 - Rappel : une chaîne est une chaîne d'ADN si elle contient seulement les caractères "a", des "t", des "c" et des "g"

Fonctions

- Exercice :
 - Écrire une fonction appelée `est_adn`
 - qui prend comme paramètre une chaîne de caractère
 - qui retourne 1 si la chaîne est une chaîne d'ADN, et 0 sinon
 - Rappel : une chaîne est une chaîne d'ADN si elle contient seulement les caractères "a", des "t", des "c" et des "g"

```
def est_adn(chaine):  
    for base in chaine:  
        if (base != "a") and (base != "t") and (base != "c") and (base != "g"):  
            return 0  
    return 1
```

Fonctions

- Exercice :
 - Écrire une fonction appelée `demander_adn`
 - qui ne prend aucun paramètre
 - qui demande à l'utilisateur d'entrer une chaîne d'ADN
 - qui vérifie que la chaîne entrée est bien un ADN, en appelant la fonction `est_adn` ; si c'est bien un ADN :
 - afficher la longueur de cet ADN
 - retourner la chaîne d'ADN
 - sinon, recommencer depuis le début

Fonctions

- Exercice :
 - Écrire une fonction appelée `demander_adn`
 - qui ne prend aucun paramètre
 - qui demande à l'utilisateur d'entrer une chaîne d'ADN
 - qui vérifie que la chaîne entrée est bien un ADN, en appelant la fonction `est_adn` ; si c'est bien un ADN :
 - afficher la longueur de cet ADN
 - retourner la chaîne d'ADN
 - sinon, recommencer depuis le début

```
def demander_adn():  
    adn = raw_input("Entrez une chaîne d'ADN : ")  
    if est_adn(adn):  
        print "Longueur :", len(adn), "paires de base"  
        return adn  
    print "Erreur, ce n'est pas un ADN !"  
    return demander_adn()
```

fonction **récurive** :
qui s'appelle elle-même

Fonctions

- Exercice :
 - Écrire une fonction appelée `demander_adn`
 - qui ne prend aucun paramètre
 - qui demande à l'utilisateur d'entrer une chaîne d'ADN
 - qui vérifie que la chaîne entrée est bien un ADN, en appelant la fonction `est_adn` ; si c'est bien un ADN :
 - afficher la longueur de cet ADN
 - retourner la chaîne d'ADN
 - sinon, recommencer depuis le début

```
def demander_adn():
```

```
    while 1:
```

```
        adn = raw_input("Entrez une chaîne d'ADN : ")
```

```
        if est_adn(adn):
```

```
            print "Longueur :", len(adn), "paires de base"
```

```
            return adn
```

```
        print "Erreur, ce n'est pas un ADN !"
```

Fonctions

- Exercice :
 - Écrire une fonction appelée `demander_oui_non`
 - qui prend en paramètre le message affiché à l'écran
 - qui demande à l'utilisateur d'entrer «oui» ou «non»
 - qui vérifie que la chaîne entrée est bien «oui» ou «non» :
 - si c'est le cas, retourner la chaîne
 - sinon, recommencer depuis le début

```
def demander_oui_non(message):  
    reponse = raw_input(message)  
    if (reponse == "oui") or (reponse == "non"):  
        return reponse  
    else:  
        print "Erreur !"  
        return demander_oui_non(message)
```

Fonctions

- Exercice :
 - Écrire une fonction appelée `poids_moleculaire_ADN` qui calcule le poids moléculaire d'une chaîne d'ADN
 - Sachant que le poids d'un couple AT est d'environ 260 ug/umol, et le poids d'un couple GC d'environ 245 ug/umol

Fonctions

- Exercice :
 - Écrire une fonction appelée `poids_moleculaire_ADN` qui calcule le poids moléculaire d'une chaîne d'ADN
 - Sachant que le poids d'un couple AT est d'environ 260 ug/umol, et le poids d'un couple GC d'environ 245 ug/umol

```
def poids_moleculaire_ADN(adn):  
    poids = 0.0  
    for base in adn:  
        if (base == "A") or (base == "T"): poids = poids + 260.0  
        elif (base == "C") or (base == "G"): poids = poids + 245.0  
    return poids
```

Fonctions

- Exercice :
 - Écrire une fonction appelée **ADN_le_plus_lourd** qui retourne la plus lourde chaîne d'ADN parmi une liste d'ADN
 - Quels paramètres ?
 - Que va-t-on utiliser pour calculer le poids des ADN ?

Fonctions

- Exercice :
 - Écrire une fonction appelée **ADN_le_plus_lourd** qui retourne la plus lourde chaîne d'ADN parmi une liste d'ADN
 - Quels paramètres ?
 - Que va-t-on utiliser pour calculer le poids des ADN ?

```
def ADN_le_plus_lourd(adns):  
    plus_lourd_adn = ""  
    poids_du_plus_lourd_adn = 0.0  
    for adn in adns:  
        poids = poids_moleculaire_ADN(adn)  
        if poids > poids_du_plus_lourd_adn:  
            plus_lourd_adn = adn  
            poids_du_plus_lourd_adn = poids  
    return plus_lourd_adn
```

Fonctions

- Exercice :
 - Écrire une fonction appelée `cherche_TATA` qui recherche une boîte TATA dans une chaîne d'ADN et retourne l'indice de la base où commence la boîte TATA, ou -1 s'il n'y a pas de boîte TATA

Fonctions

- Exercice :
 - Écrire une fonction appelée `cherche_TATA` qui recherche une boîte TATA dans une chaîne d'ADN et retourne l'indice de la base où commence la boîte TATA, ou -1 s'il n'y a pas de boîte TATA

```
def cherche_TATA(adn):  
    for i in range(len(adn) - 3):  
        if ((adn[i] == "T") and (adn[i + 1] == "A") and  
            (adn[i + 2] == "T") and (adn[i + 3] == "A")):  
            return i  
    return -1
```