# Perl - Partie 7
## Arguments de ligne de commande
## Fonctions (sans ou avec arguments)

**Résumé.**

**Arguments de ligne de commande.** Le tableau `@ARGV` est un tableau particulier initialisé par l'interpréteur Perl qui y place chacun des arguments de la ligne de commande.

```
Ligne de commande :
> ./exo.pl 19 8

Dans le script :
die "*** Args manquants sur la ligne de commande ***\n" if (! defined @ARGV);
($a,$b) = @ARGV;
```

**Fonctions.** Un très bref résumé pour savoir définir et appeler une fonction :

```
sub say_hello {            # définition
   print "hello world!\n";
}
...
say_hello();               # appel
```

Possibilité de valeurs de retour :

```
sub sum_nb {               # définition
   my $a = 6;         # variables locales
   my $b = 9;
   return $a + $b;    # retourne la valeur 15
}
...
$val = sum_nb();           # appel
```

Possibilité de transmettre des arguments à une fonction. Les arguments d'une fonction sont passés par valeurs et récupérés dans la variable tableau spéciale `@_`.

```
sub say {                  # définition
   print "$_[0] $_[1]!\n";
}
...
say("hello","world");      # appel
say("goodbye","cruel world");
```



**Exercice.** Reprenez le fichier `kinases_map.txt` de l'Exercice 3 de la Partie 5. Il y avait 5 questions dans cet exercice dont les solutions sont données en commentaire dans `exo3.pl`. Pour exécuter et tester chacune d'elles, il faut actuellement modifier le script en décommentant la solution voulue et en commentant les autres, ce qui est bigrement contraignant ... Nous allons voir ici quelques exemples d'utilisation des fonctions et des arguments en ligne de commande qui permettent d'améliorer cette situation. Inspirez vous *fortement* des scripts qui vous sont fournis, ainsi que du script `exo3.pl` donnant les solutions de l'Exercice 3 de la Partie 5.

1. Lisez *attentivement* le script `affiche.pl` et exécutez le avec comme argument le fichier `kinases_map.txt`. Ce script affiche, séparés par des tabulations, le "gene symbol", la "date" et le "cytogenetic location".
   Regardez en particulier les fonctions et la gestion des arguments en ligne de commande. Essayez les cas d'erreur pour les arguments.
2. Même question pour le script `chrom.pl`. Ce script affiche seulement les gènes localisés sur le chromosome dont le numéro est passé en argument.
3. Ecrivez le script `annee.pl` qui affichera seulement les gènes entrés dans la base après l'année spécifiée en argument.
4. Lisez *attentivement* le script `tout.pl` et exécutez le. Ce script permet de regrouper en un seul script les scripts `affiche.pl` et `chrom.pl`. Ainsi, on peut avoir les exécutions :

   ```
   1) ./tout.pl kinases_map.txt          ==> idem affiche.pl
   2) ./tout.pl kinases_map.txt 9        ==> idem chrom.pl
   ```

   Regardez en particulier les fonctions et la gestion des arguments en ligne de commande. Essayez les cas d'erreur pour les arguments.
5. Modifiez le script `tout.pl` pour y intégrer aussi les fonctionnalités du script `annee.pl` et avoir ainsi les exécutions suivantes :

   ```
   1) ./tout.pl kinases_map.txt          ==> idem affiche.pl
   2) ./tout.pl kinases_map.txt 9 chrom  ==> idem chrom.pl
   3) ./tout.pl kinases_map.txt 95 annee ==> idem annee.pl
   ```

   Pensez à traiter les cas d'erreur pour les arguments.