Guide de survie à Perl pour la Bioinformatique

Christine Tranchant-Dubreuil
christine.tranchant@ird.fr

François Sabot
francois.sabot@ird.fr

Pourquoi programmer en perl ?

- Pour traiter de grands volumes de données automatiquement

SouthGreen bioinformatics platform

Pourquoi programmer en perl ?

- Pour traiter de grands volumes de données automatiquement

  & en extraire l'information pertinente

SouthGreen bioinformatics platform

Pourquoi programmer en perl ?

- Pour traiter de grands volumes de données automatiquement
  - ➢ Rechercher des motifs dans des séquences

Pourquoi programmer en perl ?

- Pour traiter de grands volumes de données automatiquement
  - ➢ Rechercher des motifs dans des séquences
  - ➢ Lancer automatiquement une analyse sur plusieurs fichiers et analyser le résultat

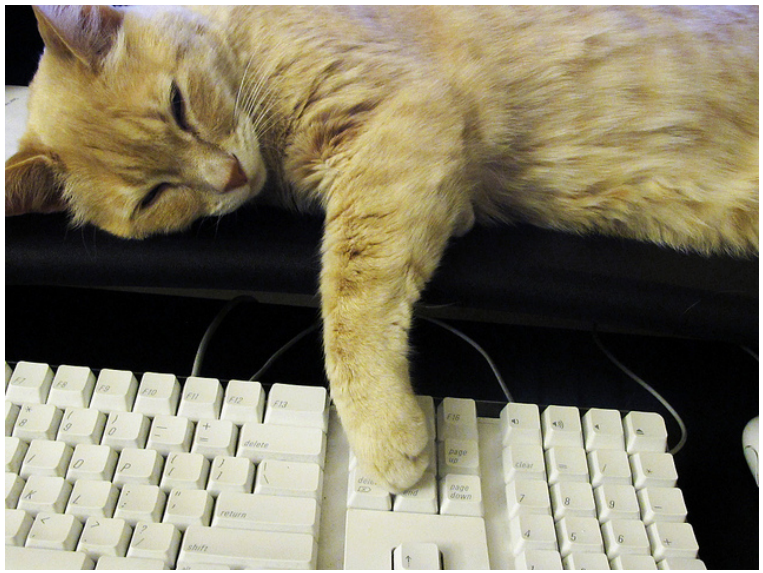
Pourquoi programmer en perl ?

- Pour traiter de grands volumes de données automatiquement
- Parcourir et transformer des fichiers - ***« parser »***

SouthGreen bioinformatics platform

Pourquoi programmer en perl ?

- Pour traiter de grands volumes de données automatiquement
- Parcourir et transformer des fichiers - ***« parser »***
- Travailler moins …



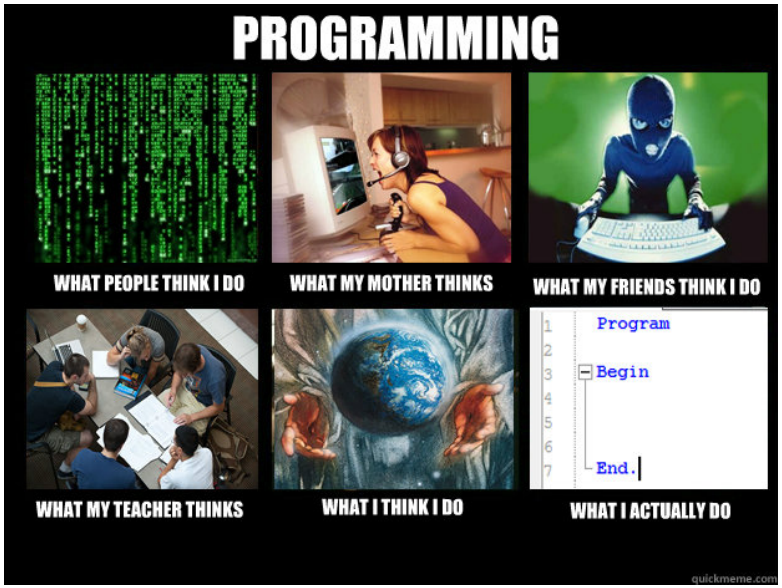
ou être plus productif

Une des qualités du programmeur est la paresse.
Larry Wall

SouthGreen bioinformatics platform

Objectifs de la formation

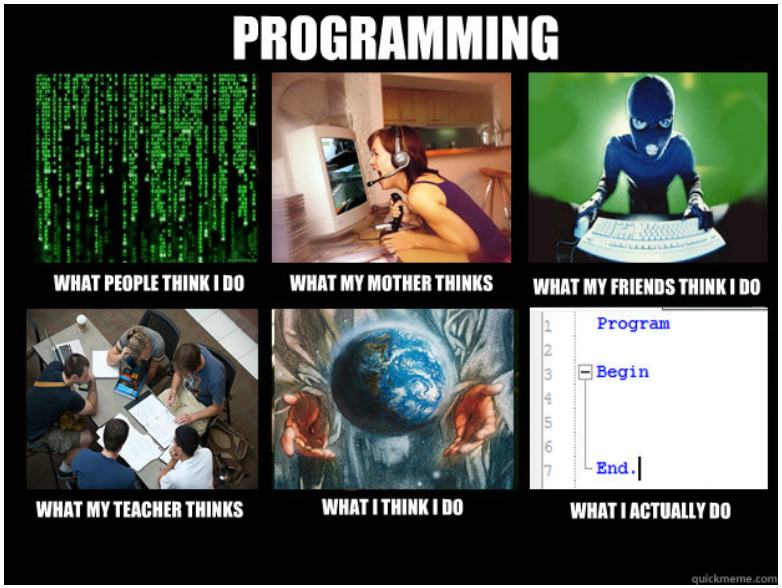
- Savoir suffisamment programmer en perl pour créer rapidement des scripts simples et faciliter l'analyse de vos données



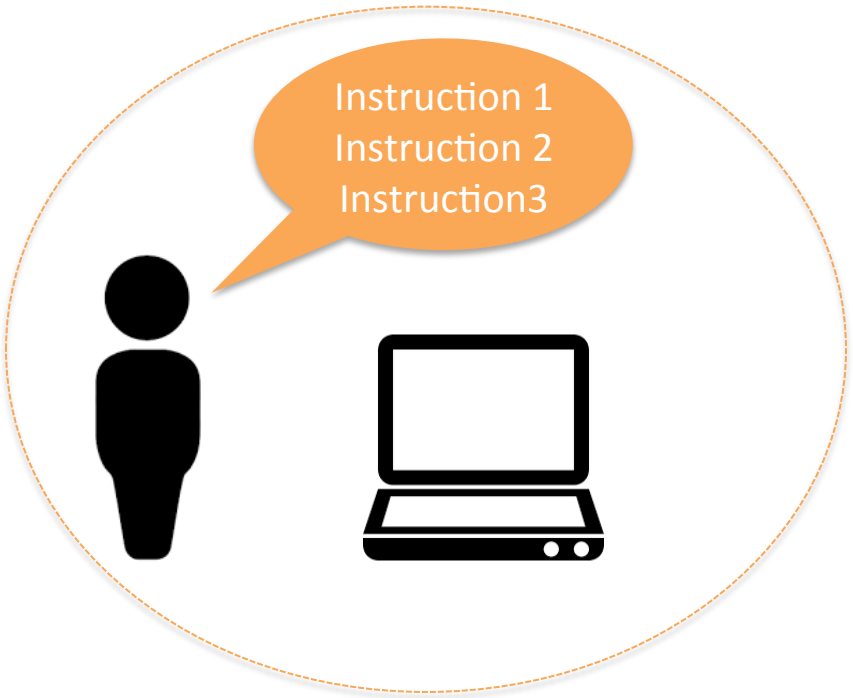
Objectifs de la formation

- Savoir suffisamment programmer en perl pour créer rapidement des scripts simples et faciliter l'analyse de vos données

Ecrire ses propres programmes une fois pour les utiliser plusieurs fois par la suite

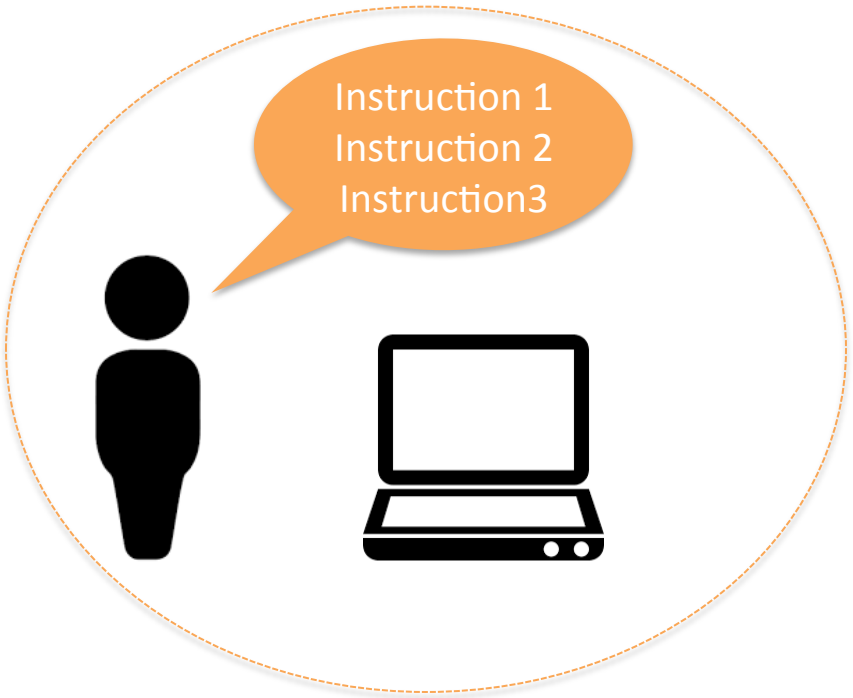


Quelques définitions



Coder…

Quelques définitions



Coder…

Dire à l'ordinateur ce qu'il doit faire exactement…

en lui donnant étape par étape une liste d'instructions à suivre

SouthGreen bioinformatics platform

Quelques définitions

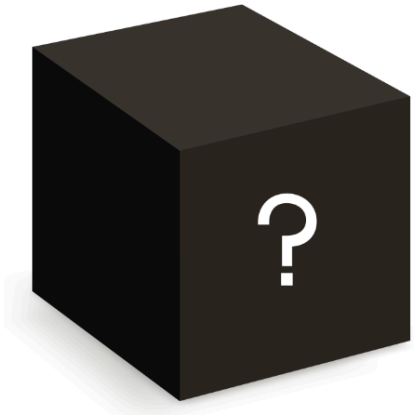
Un programme ...

SouthGreen bioinformatics platform

Quelques définitions

Un programme ...

Boite noire pour des utilisateurs



SouthGreen bioinformatics platform

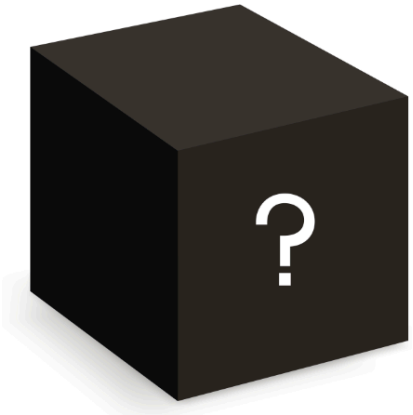
Quelques définitions

Un programme …

Boite noire pour des utilisateurs

« Input »

Données,
paramètres



SouthGreen bioinformatics platform

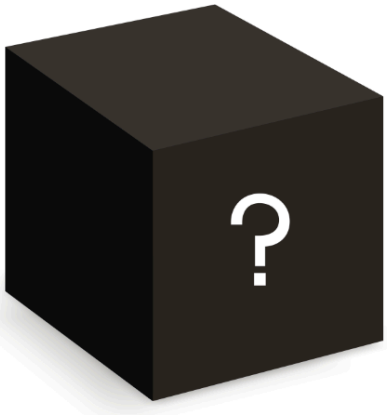
Quelques définitions

Un programme . . .

Boite noire pour des utilisateurs

« Input »

Données,
paramètres



« output »

Fichiers, images...

SouthGreen bioinformatics platform

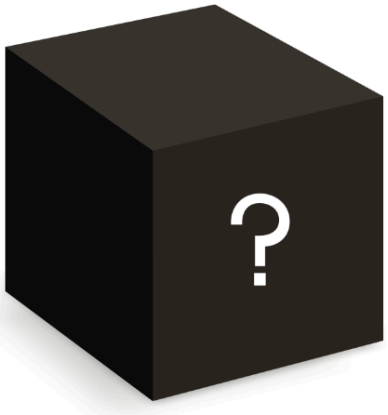
Quelques définitions

Un programme . . .

Boite noire pour des utilisateurs

« Input »

Données,
paramètres



« output »

Fichiers, images...

Mot clé : Perl

Page Google

Liste des ressources web

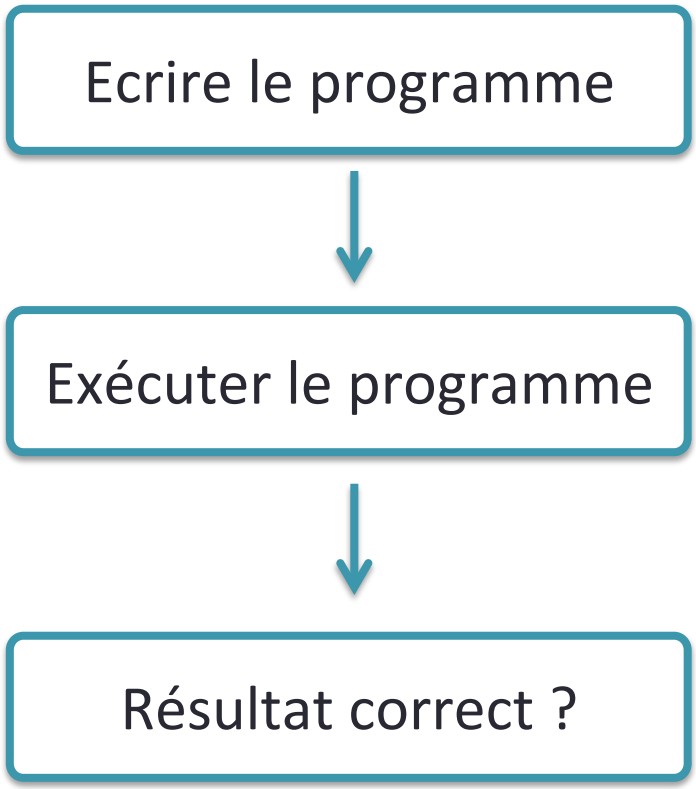
SouthGreen bioinformatics platform

Méthodologie pour coder

Ecrire le programme

SouthGreen bioinformatics platform

Méthodologie pour coder



SouthGreen bioinformatics platform

Méthodologie pour coder

Ecrire le programme

↓

Exécuter le programme

↓

Résultat correct ?

↖

Correction bugs

SouthGreen bioinformatics platform

Méthodologie pour coder

Ecrire le programme

↓

Exécuter le programme

↓

Résultat correct ?

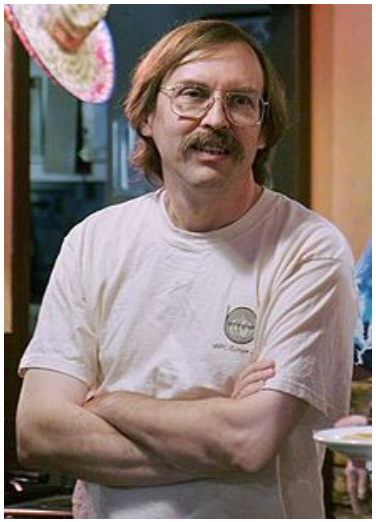
↗

Correction bugs

↖

Pourquoi Perl ?

- **P**ractical **E**xtraction and **R**eport **L**angage
- Créé en 1987 par Larry Wall
- Langage de script, syntaxe flexible, rapide à maîtriser



Larry Wall, créateur de Perl
Crédit photo : Randal Schwartz

- Particulièrement adapté à l'extraction de données de **fichiers texte** - expressions régulières

Ex : `s/\d+//g;`

SouthGreen bioinformatics platform

Pourquoi Perl ?

- Une large bibliothèque de modules, fonctions
  http://www.cpan.org
- Très présent dans la communauté bioinformatique (nombreux modules) http://www.bioperl.org
- http://www.perl.com & http://www.perl.org

Syntaxe pouvant paraître compliquée pour des nouveaux utilisateurs

There's more than one way to do it

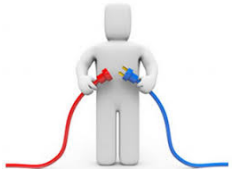
SouthGreen bioinformatics platform

Environnement de travail

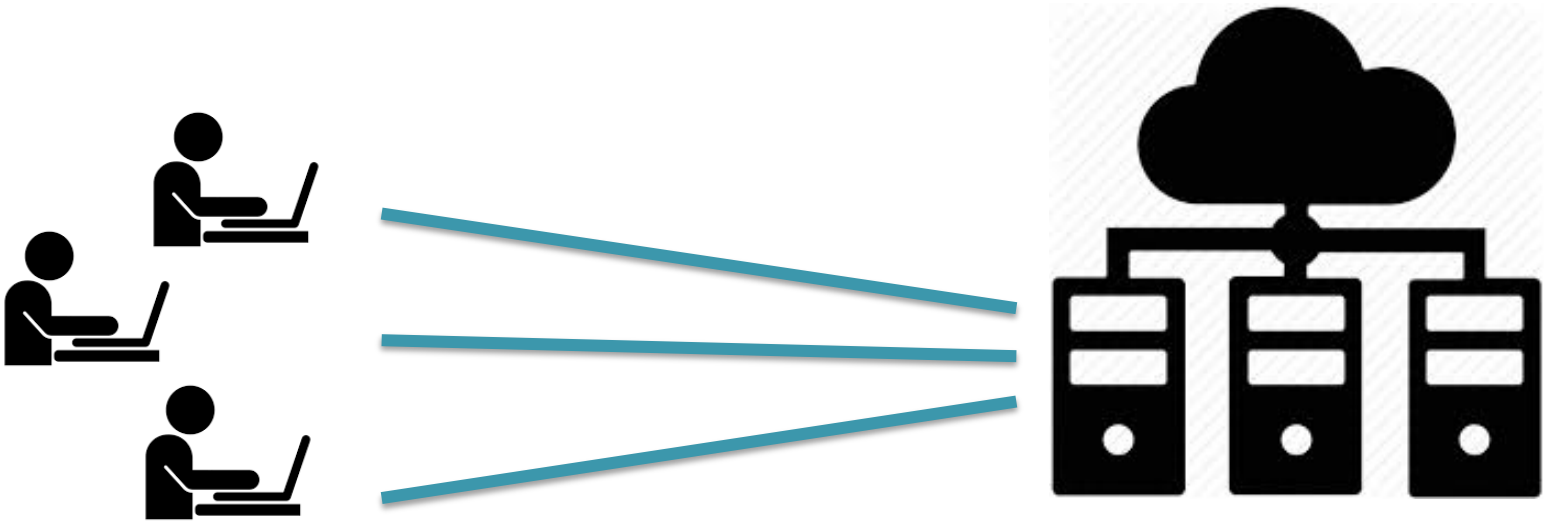
Où exécuter nos scripts perl?

Environnement de travail

Où exécuter nos scripts perl?

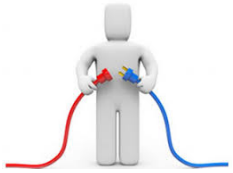


- En se connectant sur un serveur linux distant de son portable windows ou mac via le ***protocole ssh***

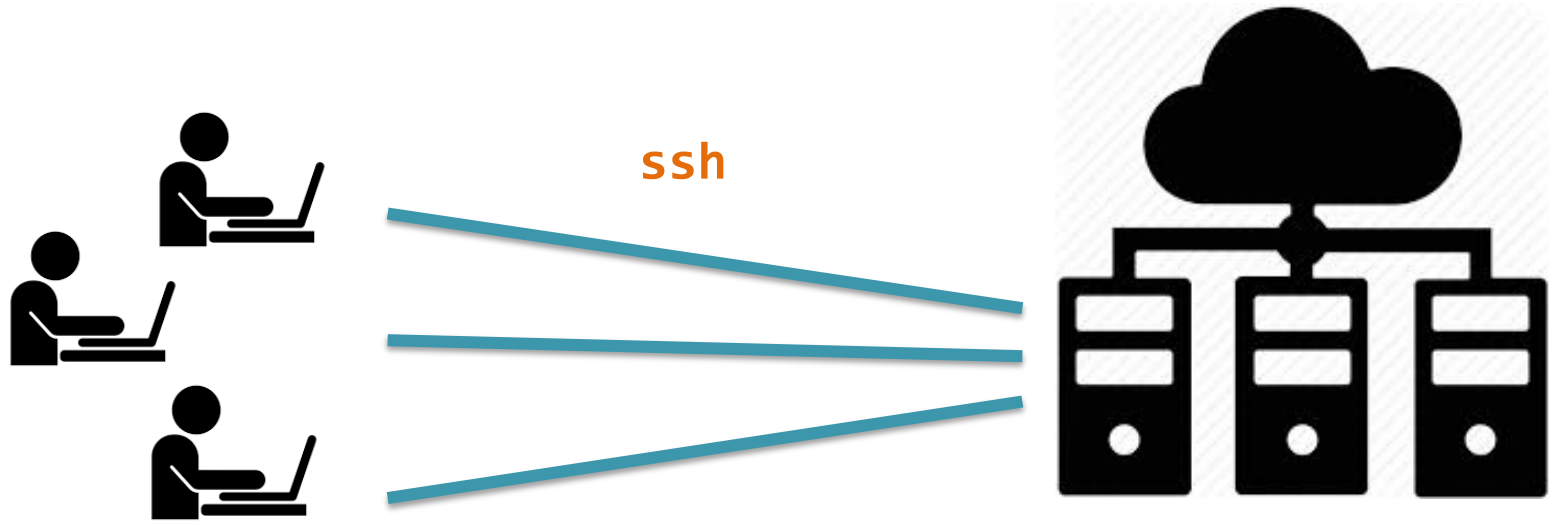


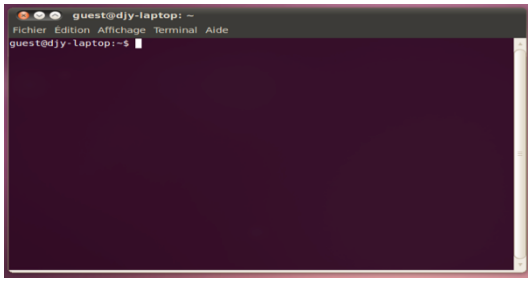
Environnement de travail

Où exécuter nos scripts perl?



- En se connectant sur un serveur linux distant de son portable windows ou mac via le ***protocole ssh***





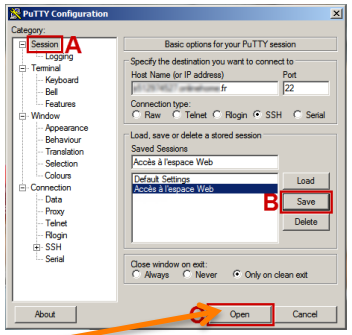
SouthGreen bioinformatics platform

Environnement de travail

Où exécuter nos scripts perl?

Sous putty :

- Host name : ***bioinfo-inter.ird.fr***
- Connection type : ***ssh***
- Saved session : ***bioinfo-inter***



N'oubliez pas de sauver votre configuration !

- Ouvrir un terminal pour se connecter en SSH
- Taper *ssh SERVER_NAME -l YOUR_LOGIN*

server_name = bioinfo-inter.ird.fr
your_login = formation1 à formation15

SouthGreen bioinformatics platform

Exécuter un script perl

```
perl nom_du_script.pl
```

SouthGreen bioinformatics platform

Exécuter un script perl

```
perl nom_du_script.pl
```

P1.1

Sous le terminal

- Vérifier que le programme perl ***affiche-v1.pl*** est dans votre *« home »*
- Exécuter le programme ***affiche-v1.pl***

SouthGreen bioinformatics platform

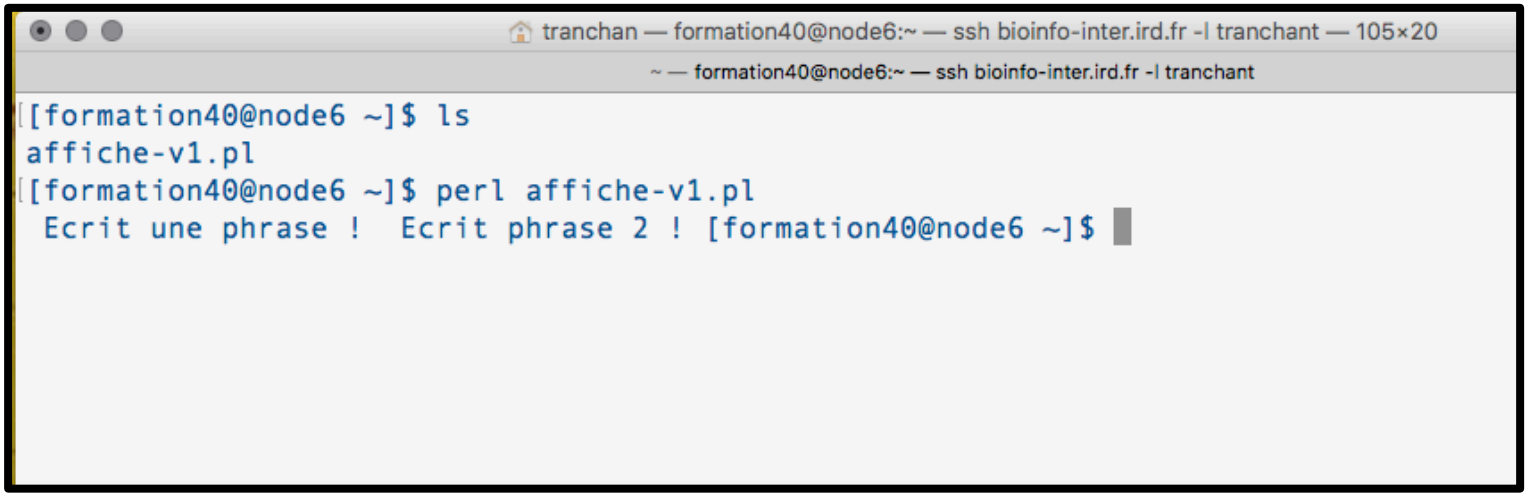
Exécuter un script perl

```
perl nom_du_script.pl
```

P1.1

Sous le terminal

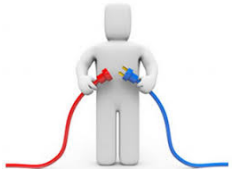
- Vérifier que le programme perl ***affiche-v1.pl*** est dans votre « *home* »
- Exécuter le programme ***affiche-v1.pl***



SouthGreen bioinformatics platform

Editer un script perl

Comment ouvrir un scripts perl sur le serveur distant ?

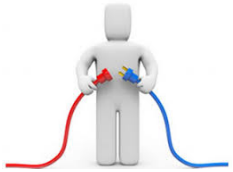


- Ouvrir directement le script perl localisé sur le serveur distant via votre éditeur de texte (Notepad++ ou KdeEdit)

SouthGreen bioinformatics platform

Editer un script perl

Comment ouvrir un scripts perl sur le serveur distant ?



- Ouvrir directement le script perl localisé sur le serveur distant via votre éditeur de texte (Notepad++ ou KdeEdit)

```
#!/usr/bin/perl

# Programme 1 : écrire un message sur le terminal

# Affiche un premier message
print " Ecrit une phrase ! ";

# Affiche un deuxième message
print  " Ecrit phrase 2 ! ";
```

affiche-v1.pl

SouthGreen bioinformatics platform

Premier script en Perl

- Toujours débuter par : `#!/usr/bin/perl -w`
- Suivis par les instructions, une instruction par ligne
- **Chaque instruction doit se terminer par** `;`
- N'hésitez pas à commenter votre script en plaçant un `#` devant votre commentaire

SouthGreen bioinformatics platform

Premier script en Perl

- Toujours débuter par : `#!/usr/bin/perl -w`
- Suivis par les instructions, une instruction par ligne
- **Chaque instruction doit se terminer par** `;`
- N'hésitez pas à commenter votre script en plaçant un `#` devant votre commentaire
  - ✓ Pour vous et vos collègues pour comprendre le code
  - ✓ Perl ignore le texte placé après un #
  - ✓ Commentaires libres

SouthGreen bioinformatics platform

Premier script en Perl

- Pas d'accent
- Premières instructions

```
print "text";          pour écrire sur la sortie (écran)
print "text \n";       pour réaliser un saut de ligne
```

SouthGreen bioinformatics platform

Modifier un script perl

P1.2

- Sauver le script ***affiche-v1.pl*** sous un nouveau nom (ex : affiche-v2.pl)
- Modifier le code de ce nouveau script en affichant d'autres textes avec ***\n***
- Exécuter ce nouveau script

SouthGreen bioinformatics platform

Modifier un script perl

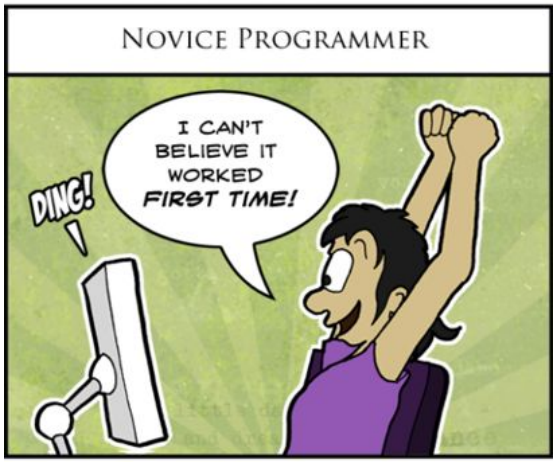
P1.3

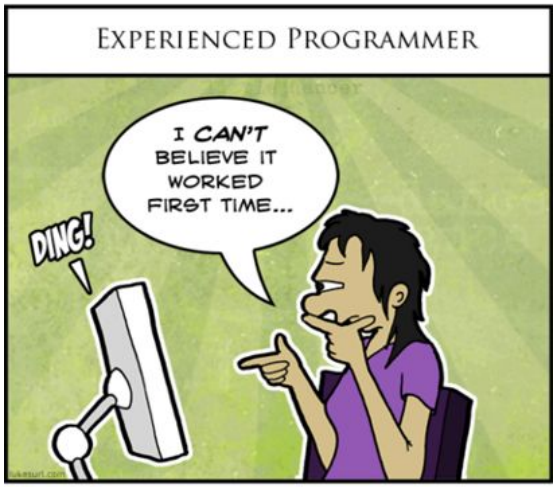
- Créer volontairement des erreurs dans votre code en retirant un ; puis un # et un “
- Observer les messages d’erreurs

Modifier un script perl

P1.3

- Créer volontairement des erreurs dans votre code en retirant un ; puis un # et un “
- Observer les messages d’erreurs





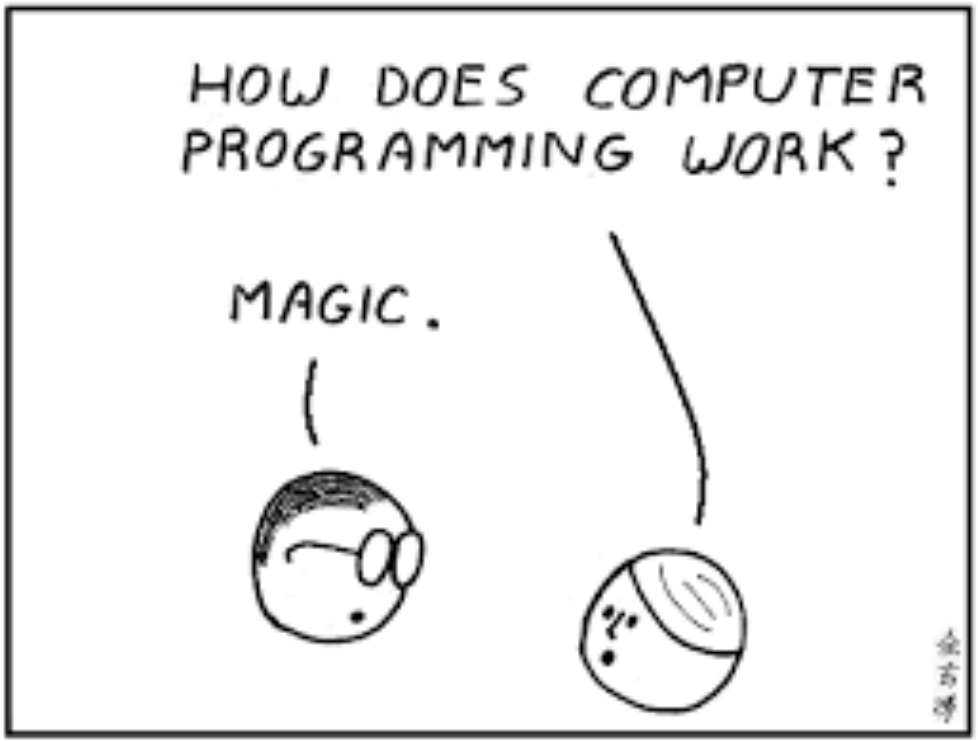
Une des principales activités du programmeur est de « débugger »…

Souvent aussi longue qu’écrire le code !

Il faut donc s’entrainer à décoder les messages d’erreurs !

SouthGreen bioinformatics platform

Perl facile, puissant et rapide*



** Variable selon les conditions expérimentales !*

SouthGreen bioinformatics platform

Perl facile*, puissant et rapide

```
#!/usr/bin/perl

# Programme 2 : télécharger la séquence au format fasta en
donnant l'accession

# Chargement du module bioperl qui va être utilisé
use Bio::DB::GenBank;

# On récupère la séquence
$gb=Bio::DB::GenBank->new();
$seq=$gb->get_Seq_by_version("CK085358.1"); #GI Number

# La séquence est sauvée dans un fichier
$out=Bio::SeqIO->new(-file => ">CK085358.1.fa");
$out->write_seq($seq);
```

retrieve-accession-v1.pl

SouthGreen bioinformatics platform

Perl en détail, c'est parti !

- Ecrire un 1er programme en perl
- Variables
- Structures de contrôle
- Lire et écrire dans un fichier
- …

SouthGreen bioinformatics platform

A la découverte des variables

SouthGreen bioinformatics platform

A la découverte des variables

Variables…

```
$nom = “perl“;

print  $nom;
```

SouthGreen bioinformatics platform

A la découverte des variables

Variables…

```
$nom = “perl“;

print  $nom;
```

« conteneur », « boîte » dans lesquels on peut stocker un objet, une information.

Cette variable va être manipulée par le programme.

SouthGreen bioinformatics platform

A la découverte des variables

Variables…

```
$nom = “perl“;

print  $nom;
```

« conteneur », « boîte » dans lesquels on peut stocker un objet, une information.

2 règles

- commencent tous par un symbole : $, @, %
- Noms de variables uniquement avec des caractères *alpha-numériques (A-Z, a-z,* 0-9) ou *underscore*
- **Sensible à la casse et pas d’espace**

SouthGreen bioinformatics platform

A la découverte des variables

Variables -> nom descriptif et court

```
$a = "ATCAGGG";                 # $a obscur
$dna_sequence_variable;         # too long
$sequence = "ATCAGGG";          # $sequence is better
$dna = "ATCAGGG";               # $dna is even better
```

SouthGreen bioinformatics platform

Les variables scalaires

SouthGreen bioinformatics platform

Variable scalaire

Variables scalaires…

```
$source = “ncbi“;
$seq_nb = 12;
```

- Les plus simples, atomiques
- Nom commençant par $
- Chaîne de caractères ou nombre

SouthGreen bioinformatics platform

Variable scalaire

Variables scalaires…

```
use strict;

my $seq_nb;
my $source = “ncbi“;
$seq_nb = 12;
```

- `use strict` pour forcer la déclaration de variable
- `my` pour déclarer une variable avant de l’utiliser
- `=` pour affecter une valeur à une variable

SouthGreen bioinformatics platform

Variable scalaire de type string

```
#!/usr/bin/perl

use strict;
use warnings;

my $number= "12";

my $composite= "There are $number sequences\n";
print $composite;
```

scalar-string-v1.pl

P2.1

- Lire le script scalar-string-v1.pl puis l'exécuter
- Créer volontairement une erreur dans votre code en retirant un `my` puis un `$`
- Observer les messages d'erreurs

SouthGreen bioinformatics platform

Variable scalaire de type string

P2.2

- Sauver le script ***retrieve-accession-v1.pl*** sous un nouveau nom (ex : ***retrieve-accession-v2.pl***)
- Modifier ce nouveau programme en ajoutant une variable `$accession` et `$fastaFile` au début du script.
- Afficher sur le terminal, une fois la séquence téléchargée le message générique suivant :

La séquence CK085358.1 a été écrite dans le fichier CK085358.1.fasta.

Fonctions liées aux chaines de caractères

`substr()` extrait une partie d'une chaîne de caractère

SouthGreen bioinformatics platform

Fonctions liées au chaine de caractères

`substr()` extrait une partie d'une chaîne de caractère

Example

- Extrait 5 bases à partir de la position *`$position`* de la séquence stockée dans la variable *`$dnaSeq`*

```
$dnaSeq = ”AATTATATTACACGATCGATCGATCGATCGACTGCTGCTACGC”;
$partOfseq = substr($dnaSeq, $position, 5);
```

SouthGreen bioinformatics platform

Fonctions liées aux chaines de caractères

`substr()` extrait une partie d'une chaîne de caractère

`length()` renvoie la longueur d'une chaîne de caractère

SouthGreen bioinformatics platform

Fonctions liées aux chaines de caractères

`substr()` extrait une partie d'une chaîne de caractère

`length()` renvoie la longueur d'une chaîne de caractère

Example

- Obtenir la longueur d'une séquence nucléique stockée dans la variable $dnaSeq

```
$dnaSeq = ”AATTATATTACACGATCGATCGATCGATCGACTGCTGCTACGC”;
$dnaLength = length($dnaSeq);
print ”Taille de la séquence : $dnaLength pb”;
```

SouthGreen bioinformatics platform

Fonctions liées aux chaines de caractères

`substr()` extrait une partie d'une chaîne de caractère

`length()` renvoie la longueur d'une chaîne de caractère

`$var1.$var2` pour concaténer 2 chaînes de caractère

SouthGreen bioinformatics platform

Fonctions liées aux chaines de caractères

`substr()` extrait une partie d'une chaîne de caractère

`length()` renvoie la longueur d'une chaîne de caractère

`$var1.$var2` pour concaténer 2 chaînes de caractère

Example

```
$accession =  ">AC909991_1\n";
$sequence  =  "ATACGTCAGCTAGCTACTGCCCT\n";
$seqFasta = $accession . $sequence ;
```

SouthGreen bioinformatics platform

Variable scalaire de type string

P2.4

Dans le script ***retrieve-accession-v2.pl***, ajouter :

- une variable qui permet de mesurer la longueur de la séquence.
- une variable qui permet d'extraire les 200ères paires de base puis de la position 215 à 400.
- Afficher le contenu de ces 2 variables à l'écran.

Fonctions liées au chaine de caractères

`substr()` extrait une partie d'une chaîne de caractère

`length()` renvoie la longueur d'une chaîne de caractère

`$var1.$var2` pour concaténer 2 chaînes de caractère

`chomp()` supprime le dernier caractère si saut de ligne

Example

```
$accession =  ">AC909991_1\n";
chomp ($accession);  # $accession = ">AC909991_1"
```

SouthGreen bioinformatics platform

Fonctions liées au chaine de caractères

`substr()` extrait une partie d'une chaîne de caractère

`length()` renvoie la longueur d'une chaîne de caractère

`$var1.$var2` pour concaténer 2 chaînes de caractère

`chomp()` supprime le dernier charactère si saut de ligne

`reverse()` renvoie la chaîne dans l'ordre inverse

Example

```
$sequence  =  "ATACGTCAGCTAGCTACTGCCCT\n";
reverse($sequence); # TCCCGTCATCGATCGACTGCATA ;
```

SouthGreen bioinformatics platform

Variable scalaire de type string

```
use strict;
use warnings;

print "Comment vous appelez-vous ? ";
my $nom = <>; # Récupération du nom de l'utilisateur
print “Bonjour, $nom sans chomp!\n";

chomp $nom;   # Retrait du saut de ligne
print "Bonjour, $nom !\n";
```

password-v1.pl

P2.4

- Exécuter le script ***password-v1.pl***
- Afficher le nom saisi dans l’ordre inverse.

SouthGreen bioinformatics platform

Opérations avec les nombres

```
#!/usr/bin/perl

# scalar-number.pl

use strict;
use warnings;

my $x=3;
my $y=2;

print "$x + $y is “. $x + $y. "\n";
print "$x - $y is “. $x - $y. "\n";
print "$x * $y is “. $x * $y. "\n";
print "$x / $y is “. $x / $y. "\n";
```

scalar-number-v1.pl

SouthGreen bioinformatics platform

Opérateurs liés aux nombres

- `++` pour ajouter un à un nombre

```
$seqNumber = 1 ;
$seqNumber ++ ; # soit égal à 2
```

⇔ `$seqNumber=$seqNumber+1`

- `--` : pour enlever un à un nombre

```
$seqNumber = 2 ;
$seqNumber -- ; # soit égal à 1
```

⇔ `$seqNumber=$seqNumber-1`

SouthGreen bioinformatics platform

Variable scalaire de type string

P2.5

- Sauver le script ***retrieve-accession-v2.pl*** sous un nouveau nom (ex : retrieve-accession-v3.pl)
- Dans ce nouveau script :
  - Récupérer aussi la séquence AK065885.1
  - Calculer le nombre de base total des 2 séquences
  - Calculer la longueur moyenne des 2 séquences

```
my $average = ($x + $y) / 2;
```

SouthGreen bioinformatics platform

Les branchements conditionnels

SouthGreen bioinformatics platform

IF – ELSE - THEN

Instruction conditionnelle…

- permet d'effectuer différentes calculs/actions en fonction de l'évaluation d'une condition booléenne

```
if (condition1 is TRUE)
{
   do this;
}
else {do wathever;}
```

SouthGreen bioinformatics platform

IF – ELSE - THEN

Instruction conditionnelle…

- permet d'effectuer différentes calculs/actions en fonction de l'évaluation d'une condition booléenne

```
if (condition1 is TRUE)
{
   do this;
}
elsif (condition2 is TRUE)
{
   do that
}
else {do wathever;}
```

SouthGreen bioinformatics platform

Opérateur de comparaison

Nombres

`$a == $b`	$a égal à $b
`$a != $b`	$a différent de $b
`$a < $b`	$a inférieur à $b
`$a > $b`	$a supérieur à $b
`$a <= $b`	$a inférieur ou égal à $b
`$a >= $b`	$a supérieur ou égal à $b

SouthGreen bioinformatics platform

Opérateur de comparaison

Nombres

`$a == $b`	$a égal à $b
`$a != $b`	$a différent de $b
`$a < $b`	$a inférieur à $b
`$a > $b`	$a supérieur à $b
`$a <= $b`	$a inférieur ou égal à $b
`$a >= $b`	$a supérieur ou égal à $b

== pour tester l'égalité

= utilisé pour l'assignation

SouthGreen bioinformatics platform

Opérateur de comparaison

Chaine de caractères

`$a eq $b`	$a égal à $b
`$a ne $b`	$a différent de $b

SouthGreen bioinformatics platform

If - then

P3.1

Dans le script retrieve-accession-v3.pl :

- Afficher un message différent si la séquence AK065885.1 a une longueur inférieure à 1000 pb ou supérieure à 1000 pb.
- Faites le test sur la 2ème accession

SouthGreen bioinformatics platform

Opérateurs /booléens logiques

OR

- Pour vérifier si au moins 1 des propositions est vraie

Il pleut	Il neige	Pleut ou Neige
FAUX	FAUX	FAUX
FAUX	VRAI	VRAI
VRAI	FAUX	VRAI
VRAI	VRAI	VRAI

SouthGreen bioinformatics platform

Opérateurs /booléens logiques

OR

- Pour vérifier si au moins 1 des propositions est vraie

Il pleut	Il neige	Pleut ou Neige
FAUX	FAUX	FAUX
FAUX	VRAI	VRAI
VRAI	FAUX	VRAI
VRAI	VRAI	VRAI

|| or

```
if ($seqLength < 100 or $seqLength > 10000)
{
    print " La séquence n'est pas traitée car sa taille est trop petite ou trop grande ! \n ";
}
```

SouthGreen bioinformatics platform

Opérateurs /booléens logiques

AND

- Pour vérifier que 2 propositions sont vraies à la fois

Beau	Intelligent	Beau ET Intelligent
FAUX	FAUX	FAUX
FAUX	VRAI	FAUX
VRAI	FAUX	FAUX
VRAI	VRAI	VRAI

SouthGreen bioinformatics platform

Opérateurs /booléens logiques

AND

- Pour vérifier que 2 propositions sont vraies à la fois

Beau	Intelligent	Beau ET Intelligent
FAUX	FAUX	FAUX
FAUX	VRAI	FAUX
VRAI	FAUX	FAUX
VRAI	VRAI	VRAI

&&

and

```
if ($seqLength > 1000 && $seqLength < 10000)
{
    print " La séquence a une taille comprise
                              entre 1000 et 10000 pb ";
}
```

SouthGreen bioinformatics platform

If - then

P3.2

Dans le script ***password-v1.pl*** :

- Demander à l'utilisateur d'entrer son mot de passe
- Si le mot de passe est « s'il te plait », le programme accueille l'utilisateur

```
Comment vous appellez-vous? Personne
Bonjour personne ! Entrez le mot de passe : s'il te plait
Accès autorisé, BIENVENUE !
```

- Sinon un message d'erreur est affiché

```
Entrez le mot de passe : sesame
Accès refusé.
```

Expression régulière...

Juste une intro

SouthGreen bioinformatics platform

Expression régulière ou rationnelle

Regular expression...

- Pour manipuler le texte de façon précise et très puissante
- 2 types de fonctionnalités : match & substitution

SouthGreen bioinformatics platform

Expression régulière ou rationnelle

Correspondance

=> Réponse : **match**

SouthGreen bioinformatics platform

Expression régulière ou rationnelle

Correspondance

=> Réponse : **match**

La variable $v commence-t-elle par un chiffre ?

`m/motif/`

SouthGreen bioinformatics platform

Expression régulière ou rationnelle

Correspondance

=> Réponse : **match**

Correspondance
+ **transformation**
=> Réponse : **substitution**

La variable $v commence-t-elle par un chiffre ?

`m/motif/`

SouthGreen bioinformatics platform

Expression régulière ou rationnelle

Correspondance

=> Réponse : **match**

Correspondance
+ **transformation**
=> Réponse : **substitution**

La variable $v commence-t-elle par un chiffre ?

`m/motif/`

Remplacer ‘toto’ par ‘titi’?

`s/motif/motif/`

SouthGreen bioinformatics platform

Expression régulière ou rationnelle

Pour lier variable et ER

```
$v =~ m/motif/
```

La variable $v commence-t-elle par un chiffre ?

```
m/motif/
```

Remplacer ‘toto’ par ‘titi’?

```
s/motif/motif/
```

SouthGreen bioinformatics platform

Expression régulière ou rationnelle

```
if ( $sequence =~ m/GAATTC/)
{
    print "EcoRI␣site␣found\n";
}
else
{
    print "no␣EcoRI␣site␣found\n";
}
```

matching-v1.pl

SouthGreen bioinformatics platform

Expression régulière ou rationnelle

P4.1

Ecrire le script matching-v1.pl :

- Le programme doit demander à l'utilisateur de saisir une séquence d'ADN sur le terminal qu'il stocke dans une variable $sequence
- Puis teste si cette séquence contient un site EcoRI en affichant un message si le site existe ou pas.

SouthGreen bioinformatics platform

Expression régulière ou rationnelle

P4.1

- Ajouter les lignes suivantes et observer ce qui se passe si on utilise l'opérateur de substitution

```
$sequence =~ s/GAATTC/gaaTtc/;

print "$sequence\n";
```

- Ajouter les lignes suivantes et observer ce qui se passe pour la variable $sequence.

```
$sequence =~ s/A/adenine/;
print "$sequence\n";

$sequence =~ s/C//;
print "$sequence\n";
```

Expression régulière ou rationnelle

P4.2

- Avec le dernier programme, seule la première occurrence du motif a été remplacée

- Pour changer toutes les occurrences, ajouter à la fin de l'ER, la lettre g (option globale)

```
$sequence =~ s/C//g;
```

SouthGreen bioinformatics platform

Expression régulière ou rationnelle

P4.3

Ajouter les lignes suivantes au script et observer ce qui se passe quand :

- à la fin de l'ER, il n'y a pas la lettre i
- puis quand la lettre i est ajoutée

```
my $protein = "←
MVGGKKKTKICDKVSHEEDRISQLPEPLISEILFHLSTKDLWQSVPGL";

print "Protein␣contains␣proline\n" if ($protein
=~ /p/i);
```

SouthGreen bioinformatics platform

Expression régulière ou rationnelle

P4.4

- L'opérateur de substitution retourne le nombre de changements réalisés. Ajouter les lignes suivantes à votre script :

```
my $sequence = "AACTAGCGGAATTCCGACCGT" ;
my $g_count = ($sequence =~ s/G/G/);
print "The␣letter␣G␣occurs␣$g_count␣times␣in␣←
$sequence\n »;
```

SouthGreen bioinformatics platform

Opérateurs des ER

```
$seq =~ m/GAATTC/        match
$seq !~ m/GAATTC/        no match
```

```
$seq =~ s/A/U/           substitution
```

SouthGreen bioinformatics platform

Exercice : Composition ADN

- Votre programme devra demander à l'utilisateur de saisir une séquence, la stocker dans une variable et afficher les informations suivantes :
  - Taille de la séquence en paire de base
  - Nombre de nucléotides A, C, G et T
  - Le pourcentage de chaque nucléotide
  - La fraction GC
  - La traduction

Les variables de type tableau

SouthGreen bioinformatics platform

Variable de type tableau

Variables tableau...

```
@gene = (’CK085358.1’,’CK085357.1’,
CK085356.1’);
```

- Listes de scalaires
- Nom commençant par @

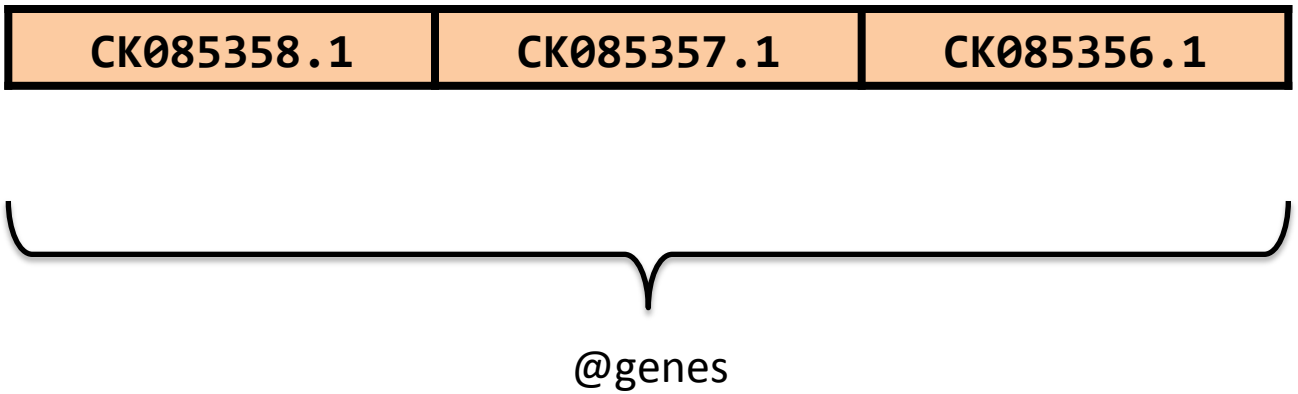
SouthGreen bioinformatics platform

Variable de type tableau

Variables tableau...

```
@gene = (’CK085358.1’,’CK085357.1’,
CK085356.1’);
```

- Listes de scalaires
- Nom commençant par @



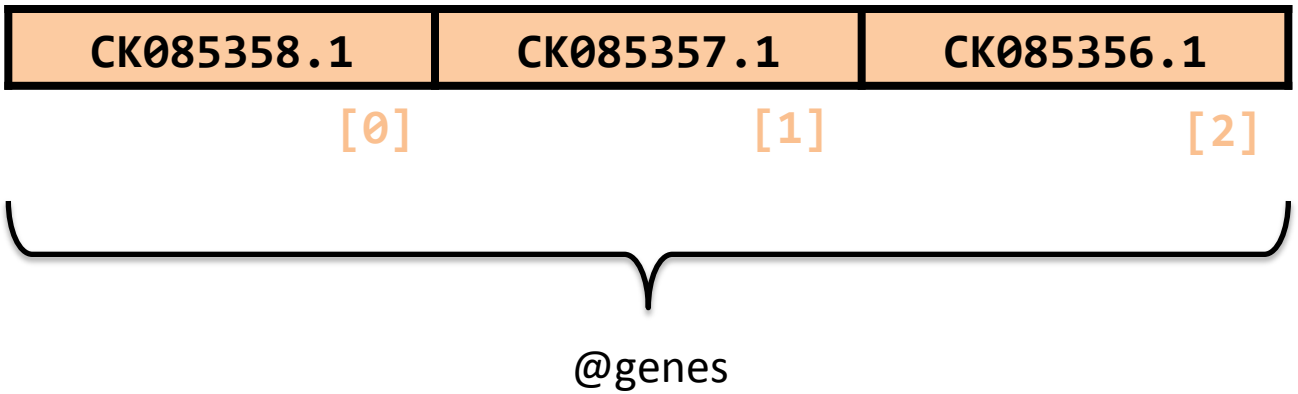
SouthGreen bioinformatics platform

Variable de type tableau

Variables tableau…

```
@gene = (’CK085358.1’,’CK085357.1’,
CK085356.1’);
```

- Listes de scalaires
- Nom commençant par @
- Chaque élément de la liste = variable scalaire



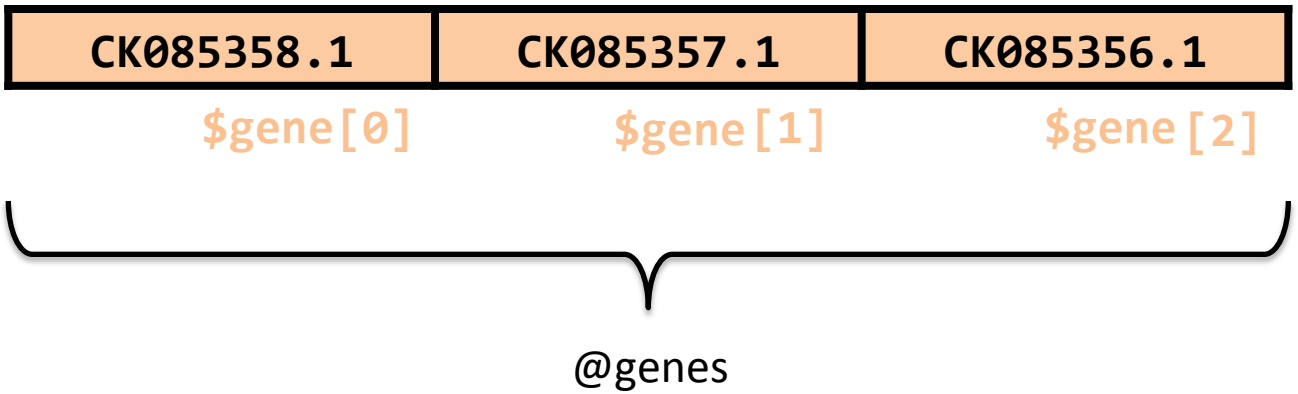
SouthGreen bioinformatics platform

Variable de type tableau

Variables tableau...

```
@gene = (’CK085358.1’,’CK085357.1’,
CK085356.1’);
print "$gene[1]\n"; # CK085357.1
print "$gene[0]\n"; # CK085358.1
```

- Listes de scalaires
- Nom commençant par @
- Chaque élément de la liste = variable scalaire



SouthGreen bioinformatics platform

Variable de type tableau

```
#!/usr/bin/perl

my @animals = (’cat’, ’dog’, ’pig’);

print "1st␣animal␣in␣array␣is:␣$animals[0]\n";
print "2nd␣animal␣in␣array␣is:␣$animals[1]\n";
print "Entire␣animals␣array␣contains :@animals\n";
```

animals-v1.pl

P5.1

- Créer et exécuter le programme ci-dessus
- Créer volontairement une erreur dans votre code comme ci dessous et observer les messages d’erreurs:

```
print "@animals[0]\n";
```

SouthGreen bioinformatics platform

Variable de type tableau

Variables tableau…

```
@gene = (’CK085358.1’,’CK085357.1’,
CK085356.1’);
print "$gene[1]\n"; # CK085357.1
print "$gene[0]\n"; # CK085358.1
```

- Listes de scalaires
- Nom commençant par @
- Chaque élèment de la liste = variable scalaire

Propriétés

- Ils sont dynamiques
- On peut facilement ajouter ou retirer des valeurs de la liste

SouthGreen bioinformatics platform

Fonctions liées aux tableaux

`push()`	ajouter une valeur à la fin de la liste
`pop()`	enlever la dernière valeur de la liste
`shift()`	enlever la première valeur de la liste

Example

```
push(@gene,"AKYHIA");
my $geneExtrait = pop(@gene);
$geneExtrait = shift(@gene);
```

SouthGreen bioinformatics platform

Variable de type tableau

```
my $length=@animals;
print "Entire␣animals␣array $length
elements : @animals\n";

# Add lines with pop, push & shift functions
and print your array after every changment
```

animals-v2.pl

P5.2

- Modifier le programme animals-v1.pl en modifiant le tableau avec les fonctions pop, push & shift
- Stocker les valeurs retirées du tableau avec les fonction pop et shift dans une variable

SouthGreen bioinformatics platform

Fonctions liées aux tableaux

`join(separateur,@list)` concatène les valeurs d'un tableau

```
#!/usr/bin/perl

use warnings ;
use strict;

my @geneNames= ("unc-10","cyc-1","act-1","let-7”,"dyf-2”);

my $joined_string = join (“ : ", @gene_names) ;
print "$joined_string\n";
```

String-array-v1.pl

SouthGreen bioinformatics platform

Fonctions liées aux tableaux

`join(separateur,@list)` concatène les valeurs d'un tableau

`split(separateur,$text)` découpe une chaine en tableau

```
#!/usr/bin/perl

use warnings ;
use strict;

my $geneList= ("unc-10;cyc-1;act-1;let-7;dyf-2”);

my @geneNames= split /;/, $geneList;
```

string-array-v2.pl

SouthGreen bioinformatics platform

Variable de type tableau

P5.3

- Exécuter le programme string-array-v1.pl
- Ajouter les lignes ci-dessous et utiliser la fonction split pour digérer une séquence ADN par l'enzyme EcoRI. Afficher le résultat sur l'écran.

```
my $dna = "aaaaGAATTCttttttGAATTCgggggggg" ;
my $EcoRI = "GAATTC";
```

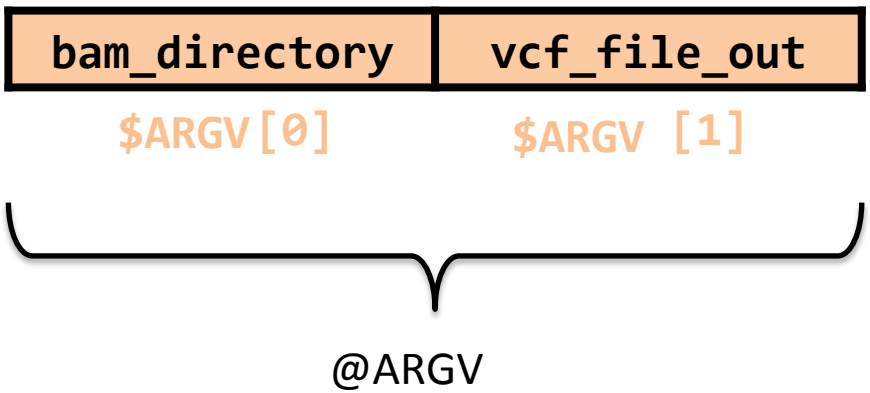
SouthGreen bioinformatics platform

@ARGV

@ARGV... tableau réservé

pour passer des arguments

```
perl detect_SNP.pl bam_directory vcf_file_out
```



SouthGreen bioinformatics platform

Ajouter des arguments à un script

P5.6

Au niveau du script retrieve-accession-v3.pl

- Modifier le afin que les 2 accessions utilisées dans le programme soient donnés en argument lors du lancement du programme.
- Sauver ce script sous un nouveau nom : ***retrieve-accession-arg.pl***

Fonctions liées aux tableaux

Pour trier les valeurs

=> fonction `sort()` + opérateurs de comparaison `cmp` et `<=>`

Chaîne de caractère / ordre lexical

```
sort ({$a cmp $b} @list ) ou sort (@list )
```

Chaîne de caractère / ordre numérique

```
sort ({$a <=> $b} @list )
```

SouthGreen bioinformatics platform

Variables de type tableau

P5.4

- Créer le programme ci-dessous ***sorting-array-v1.pl*** et exécuter le

```
#!/usr/bin/perl

use strict;
use warnings;

# An unsorted list
my @list=("c","b","a","C","B","A","a","b","c",3,2,1);

my @sorted_list = sort @list ;
print "Default sorting: @sorted_list\n";
```

SouthGreen bioinformatics platform

Variable de type tableau

P5.5

- Changer le programme précédent en utilisant un **tableau de nombres** et exécuter le
- Changer le tri en utilisant l'opérateur **<=>**
- Trier dans l'ordre inverse comme ci dessous

```
@sorted_list = sort ({$b <=> $a } @list) ;
```

SouthGreen bioinformatics platform

Les structures de contrôle foreach, for, while

SouthGreen bioinformatics platform

Les boucles

Structure de contrôle…

Pour exécuter plusieurs fois les mêmes instructions sans réécrire les mêmes lignes de code

SouthGreen bioinformatics platform

Les boucles

Structure de contrôle…

Pour exécuter plusieurs fois les mêmes instructions sans réécrire les mêmes lignes de code

Différentes structures

- foreach
- while
- for
- …

SouthGreen bioinformatics platform

foreach

foreach...

- Pour parcourir une liste
- Exécuter les mêmes instructions sur chaque élément de la liste

```
foreach variable (liste)
{
   instruction1;
   instruction2;
}
```

SouthGreen bioinformatics platform

Variable scalaire de type string

```
foreach my $element (@animals)
{
    print $element;
}
```

P6.1

- Modifier le programme animals-v1.pl en ajoutant une boucle foreach

SouthGreen bioinformatics platform

Modifier un script perl

P6.2

Dans le script ***retrieve-accession-arg.pl*** :

- Optimiser le code en utilisant une boucle foreach
- Afficher la longueur en pb pour chaque séquence
- Puis calculer la taille moyenne
- Sauver ce script sous un nouveau nom :

retrieve-accession-arg-v2.pl

SouthGreen bioinformatics platform

while

while...

- Les instructions sont exécutées tant que la condition est vraie

```
while(liste)
{
   intruction1;
   instruction2;
}
```

SouthGreen bioinformatics platform

Modifier un script perl

P6.3

Au niveau du script ***password-v1.pl***,

- Demander à l'utilisateur d'entrer son mot de passe jusqu'à ce que le mot de passe saisi soit correct. Sauver ce script sous le nom de ***password-v2.pl***
- Puis modifier le en permettant à l'utilisateur de saisir 3 fois uniquement son mot de passe

rq : on va introduire une variable qui va compter le nombre de tentative

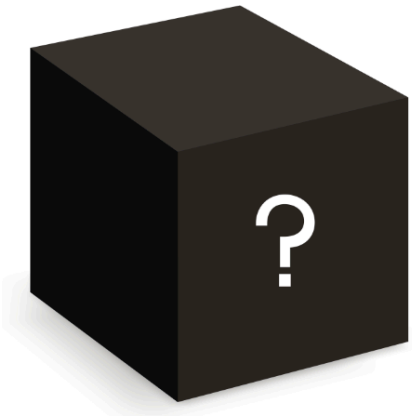
Entrée / sortie

SouthGreen bioinformatics platform

Entrée / sortie

« Input »

Données, paramètres



« output »

Fichiers, images...

SouthGreen bioinformatics platform

Ouvrir un fichier

fonction open()…

```
open (descripteur, "mode", nom_u_fichier);
```

identifiant du fichier
après ouverture (variable)

Mode d'ouverture

<	**lecture**	**open ($fh, "<", $fileName);**
>	Écriture (écrasement)	open ($fh, ">", $fileName);
>>	ajout	open ($fh, ">>", $fileName);

SouthGreen bioinformatics platform

Ouvrir un fichier

fonction open() avec fonction die…

```
open(descripteur,  "mode", nom_du_fichier)
      or die "Impossible d'ouvrir le fichier ! ";
```

- Si erreur, met fin au programme en affichant le message

SouthGreen bioinformatics platform

Fermer un fichier

fonction close()

```
close(descripteur);
```

SouthGreen bioinformatics platform

Lire un fichier

Avec l'opérateur chevron...

- On lit la prochaine ligne et la stocke dans une variable scalaire

```
my $line = <$fd>;
```

- On stocke toutes les lignes dans un tableau

```
my @line = <$fd>;
```

SouthGreen bioinformatics platform

Lire un fichier

Avec l'opérateur chevron...

- On lit ligne à ligne avec une boucle ***while***

```
while (my $line = <$fd>)
{
    chomp $ligne;
    print « Ligne lue : $line »;
}
```

SouthGreen bioinformatics platform

Modifier un script perl

P7.1

Ecrire un script ***retrieve-accession-fichier.pl*** :

- Un nom de fichier est donné par l'utilisateur en argument. Ce fichier contient une liste d'accession.
- Ce fichier va être lu par le programme ; à la lecture de chaque accession, la séquence sera récupérée et sauvée dans un fichier fasta.

La variable de type hash

SouthGreen bioinformatics platform

Variable de type hash

Variables hash…

```
my %genetic_code = (
    'ATG' => 'Met',
    'AAA' => 'Lys',
    'CCA' => 'Pro'
);
```

- Nom commençant par %
- Similaire au tableau
- Listes de scalaires
- Indexés par des chaines de caractère et pas des nombres

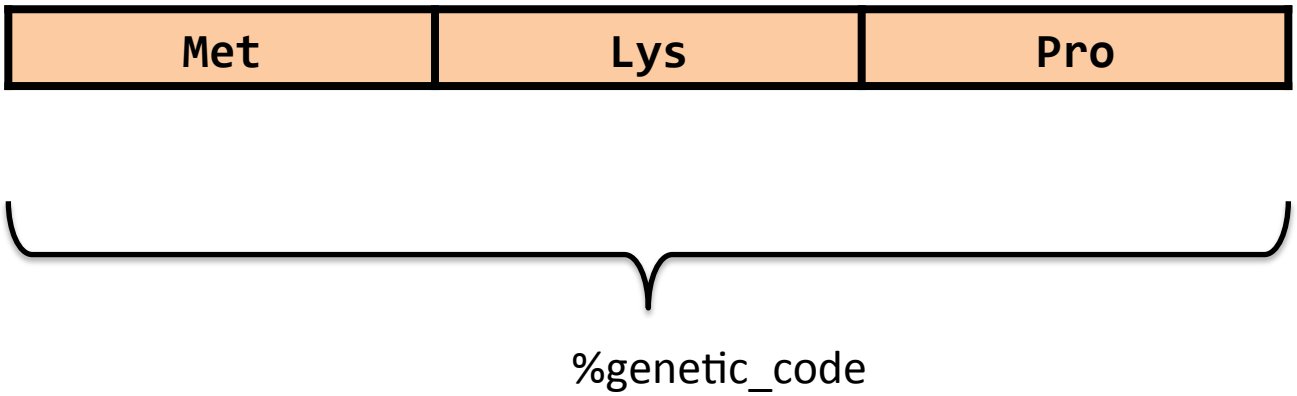
SouthGreen bioinformatics platform

Variable de type hash

Variables hash…

```
my %genetic_code = (
    'ATG' => 'Met',
    'AAA' => 'Lys',
    'CCA' => 'Pro'
);
```

- Listes de scalaires
- Nom commençant par %
- Indexés par des chaines de caractère et pas des nombres



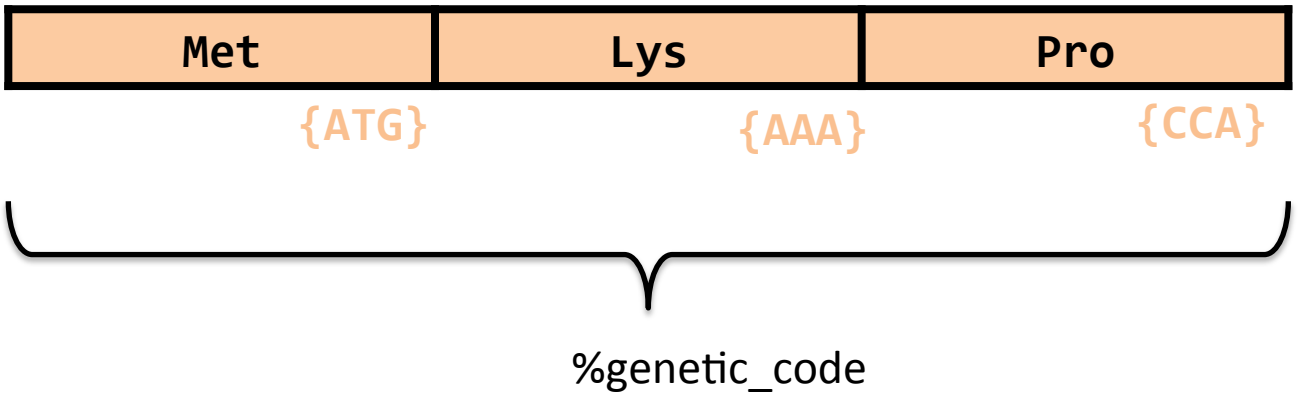
SouthGreen bioinformatics platform

Variable de type hash

Variables hash...

```
my %genetic_code = (
    'ATG' => 'Met',
    'AAA' => 'Lys',
    'CCA' => 'Pro'
);
```

- Listes de scalaires
- Nom commençant par %
- Indexés par des chaines de caractère et pas des nombres



SouthGreen bioinformatics platform

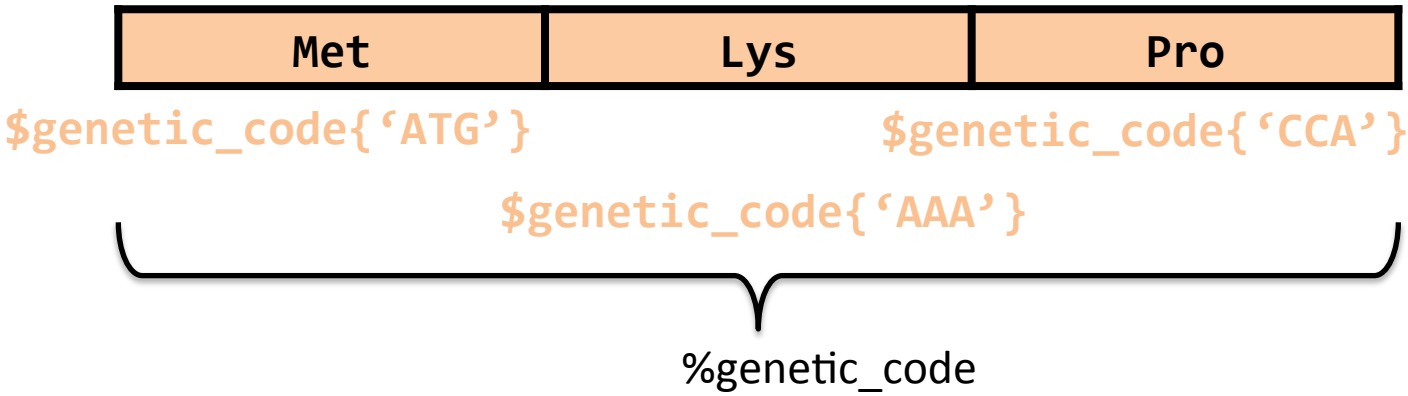
Variable de type hash

Variables hash...

```
my %genetic_code = (
    'ATG' => 'Met',
    'AAA' => 'Lys',
    'CCA' => 'Pro'
);

print $genetic_code{'ATG'};
```

- Listes de scalaires
- Nom commençant par %
- Indexés par des chaines de caractère et pas des nombres



Fonctions liées aux hash

`keys(%hash)` retourne une liste des clés

Example

```
my @cles = keys(%genetic_code);
```

SouthGreen bioinformatics platform

Fonctions liées aux « hash »

`keys(%hash)` retourne une liste des clés

Example

```
foreach my $cle (keys(%genetic_code))
{
  print « Triplet $cle code pour $genetic_code{‘$cle’};»
}
```

SouthGreen bioinformatics platform

Fonctions liées aux hash

`keys(%hash)` retourne une liste des clés

`values(%hash)` retourne une liste des valeurs de la liste

Example

```
foreach my $valeur (values(%genetic_code))
{
  print ” Valeur : $valeur\n”;
}
```

SouthGreen bioinformatics platform

Variable de type hash

```
my %genetic_code =    (
                “ATG” => “Met”,
                “AAA” => “Lys”,
                “CCA” => “Pro”    );

print "$genetic_code{“ATG”}\n";

foreach my $key ( keys %genetic_code )
{
    print "$key - $genetic_code{$key}\n";
}
```

hash-v1.pl

P8.1

- Créer et exécuter le programme ci-dessus

SouthGreen bioinformatics platform

Insérer, supprimer et tester

- Ajouter 2 valeurs à un hash

```
$genetic_code{“CCG”} = “Pro”;
$genetic_code{“AAA”} = “Lysine”;
```

SouthGreen bioinformatics platform

Insérer, supprimer et test

- Ajouter 2 valeurs à un hash

```
$genetic_code{“CCG”} = “Pro”;
$genetic_code{“AAA”} = “Lysine”;
```

- Supprimer une valeur de la table de hachage :

```
delete $genetic_code{“AAA”};
```

SouthGreen bioinformatics platform

Insérer, supprimer et test

- Tester si une valeur existe

```
if (exists $genetic_code{“AAA”})
{
   print "AAA␣codon␣has␣a␣value:
                $genetic_code{‘AAA’}\n";
}
else
{
   print "No value set for AAA codon\n";
}
```

SouthGreen bioinformatics platform

Exercice

- Votre programme devra analyser une séquence donnée par l'utilisateur comme argument du programme et afficher le « codon usage » pour une séquence.

- Comment écrire ce programme ?
  - Quel input ?
  - Quel output ?
  - Quelles variables utilisées ? Scalaires, hash
  - Listes des principales opérations et instructions

Pour perl, la pensée magique ne fonctionne pas !

- Il faut pratiquer !!!!
- *Restez calme et à vous de jouer !*

Le matériel pédagogique utilisé pour ces enseignements est mis à disposition selon les termes de la licence Creative Commons Attribution - Pas d'Utilisation Commerciale - Partage dans les Mêmes Conditions (BY-NC-SA) 4.0 International: http://creativecommons.org/licenses/by-nc-sa/4.0/