MBANI-TALAMESSO Lise
MADIKITA BOKATOLA Divine
PRAGASSAM Anthony
CHERIF Aimen

L2-SDV-G4
ISV43

Mode d'emploi programme Enzyme de restriction

Ce programme permet à l'utilisateur de faire une digestion enzymatique (de couper) d'une séquence d'ADN par une ou plusieurs enzymes de restriction.

Il est constitué d'un menu principal comportant plusieurs options :

1. Choix de l'enzyme de restriction
2. Séquence d'ADN (Fasta)
3. Forme de la séquence
4. Exécution
0. Quit

Ces différentes options s'affichent au lancement du programme et l'utilisateur pourra commencer par l'option de son choix. Nous vous conseillons par ailleurs de commencer par l'une des deux premières options pour que le programme initialise votre choix.

Guide de l'utilisateur

Pour utiliser le programme, l'utilisateur devra entrer le chiffre correspondant à l'option de son choix :

1. Choix de l'enzyme : Option permettant à l'utilisateur d'entrer l'enzyme de son choix tout en respectant la nomenclature conventionnelle des enzymes de restriction :

   - La première lettre est en majuscule cela correspond à la première lettre du genre de la bactérie
   - La seconde et troisième lettre sont en minuscule et correspondent aux deux premières lettres de l'espèce bactérienne
   - La quatrième lettre correspond à la souche bactérienne, elle est écrite en majuscule mais n'est pas présente dans toutes les enzymes de restriction
   - Enfin un chiffre romain donne le numéro d'ordre de caractérisation de l'enzyme

Exemple : **EcoRI :** Enzyme de restriction d'**E**scherichia (genre) **co**li (espèce) souche **R**Y317 la première à être caractérisée (**I**)
L'utilisateur peut choisir jusqu'à 3 enzymes de restriction de son choix, pour cela, s'affichera à chaque choix de l'option 1, une autre lui précisant qu'il peut encore entrer une enzyme. Pour modifier le choix d'une enzyme entrée et validée, l'utilisateur devra utiliser l'option « Quit », qui entraîne la perte de toutes les données entrées, l'utilisateur devra dans ce cas, tout recommencer.

2. Séquence d'ADN (Fasta) : Option permettant à l'utilisateur d'entrer une séquence d'ADN de taille maximal de 100000000 caractères au format fasta.
3. Forme de la séquence : Option permettant d'entrer la forme de la séquence : linéaire ou circulaire tout en respectant la règle de l'écriture précisée dans le programme.

4. Exécution : Option permettant à l'utilisateur d'exécuter enfin le programme après avoir entré les enzymes de son choix, la séquence à couper et la forme de la séquence. Cette option vérifiera la bonne entrée de toute les options avant exécution. En cas de mauvaise entrée, le programme le précisera à l'utilisateur à chaque étape.

0. Quit : Entraîne la perte des données entrées.

Plus l'utilisateur avance dans le choix de ces options, s'affichera une autre option :

« 9. Back(menu) » : permettant de revenir au menu principal.

Il est également possible de modifier la séquence ainsi que sa forme lorsqu'elles ont été entrées et validées en revenant sur les options « séquence d'ADN » et « Forme de la séquence ». Cette fonction n'est valable que pour ces deux options.

Lors de l'entrée de votre choix, il vous est demandé de faire attention par exemple en entrant les options qu'il faut ou encore bien écrire la séquence car lorsqu'une erreur d'écriture est répétée plus de deux fois, le programme s'arrête automatiquement cela permet d'éviter à l'utilisateur de rester bloqué dans un champ quelconque ; le programme explique néanmoins à l'utilisateur comment faire à chaque étape pour éviter ce problème, il vous suffira de bien suivre les instructions.

Lors du lancement du programme, l'utilisateur devra le placer dans un dossier dans son ordinateur, ce dossier comprendra en plus du programme : le fichier des enzymes de restriction « enzymeR2.txt » ainsi que le fichier audio « aux_portes_de_la_cite.wav » que l'utilisateur recevra car ces fichiers seront utiles au bon fonctionnement du programme.

Bonne utilisation !

MBANI-TALAMESSO Lise
MADIKITA BOKATOLA Divine
PRAGASSAM Anthony
CHERIF Aimen

L2-SDV-G4
ISV43

Algorithme du programme

Le programme a été divisé en plusieurs fonctions pour faciliter la rédaction. Pour cela, nous avons commencé par initialiser différentes variables sous forme de liste, chaîne de caractère, type booléen et entier que nous avons utilisé dans la suite du programme.

I. Menu principal (def main menu ())

Description du menu principal avec tous les champs qu'il contient.

Lors de l'ouverture du programme, cette fonction permet d'afficher ces différentes options qui constitueront le menu :

1. Choix de l'enzyme de restriction
2. Séquence d'ADN (Fasta)
3. Forme de la séquence
4. Exécution

0. Quit

L'utilisateur devra ensuite choisir l'une des différentes options en fonction de ce par quoi il veut commencer.

II. Def exec_menu(choice)

Cette fonction permettra la gestion du programme tout entier en anticipant sur les différentes erreurs que peut rencontrer le programme lors de son utilisation par l'utilisateur pour éviter les bugs. Nous avons choisi d'anticiper sur les différents problèmes qui peuvent survenir comme une commande de l'utilisateur vide ou erronée dans le choix du menu qui renvoie automatiquement au menu principal en utilisant la variable try, except. Si trois commandes erronées sont entrées à la suite, le programme s'arrête, ceci pour pallier un bug de l'entrée de séquence décrit dans le rapport de projet.

III. Choix de l'enzyme : def enzyme ()

Nous avons remis à 0 le compteur d'erreur à chaque entrée dans un menu pour limiter les cas où le programme s'arrête à cause d'erreurs de commandes répétés par l'utilisateurs et non par le bug que cette fonctionnalité est sensée pallier.

Nous avons initialisé un compteur d'erreur qui arrêtera le programme en cas d'erreur répétée et l'utilisateur pourra relancer le programme pour recommencer. Cela évitera que le programme bug.

Nous avons utilisé la variable « global » afin de pouvoir utiliser le compteur dans toutes les autres fonctions du programme cela nous évitera d'initialiser une nouvelle variable à chaque fois.

Nous avons opté pour l'utilisation d'un dictionnaire « dictcodon » afin de facilement stocker les textes dans les fichiers, qui serviront dans le programme.

Les enzymes seront entrées dans une liste, cela nous permettra de mieux les utiliser pour couper la séquence entrée.

L'utilisation d'un fichier nous a été indispensable pour stocker les différentes enzymes de restriction. Lors du choix de cette option, l'utilisateur choisira une enzyme déjà reconnu par le programme.

Nous avons également affiché une petite description de comment l'utilisateur devra entrer les noms des enzymes afin d'éviter qu'il utilise une écriture que le programme ne pourra pas reconnaitre.

IV. Séquence d'ADN : def sequence()

Nous avons encore remis à 0 le compteur de gestion d'erreur du programme « Idstop » et un nouveau compteur « pos » qui repèrera où se trouve l'erreur de l'utilisateur lors de l'entrée de sa séquence.

Nous avons ensuite choisi d'afficher différentes étapes que devra suivre l'utilisateur en lui expliquant quel format devra-t-il utiliser pour la séquence à entrer.

V. Forme : def forme()

Cette fonction permet à l'utilisateur de choisir la forme de sa séquence (linéaire ou circulaire). Elle affichera la nomenclature exacte que l'utilisateur devra utiliser. Même si ce n'est pas spécifié dès le début, pour entrer sont choix, l'utilisateur peut entrer L, l, linéaire ou lineaire pour signifier que la séquence est linéaire et C, c ou circulaire pour signifier que la séquence est circulaire, tout ça pour ajouter plus de flexibilité et de rapidité lors de l'entrée des données.

La variable « idf », permet au programmer de vérifier que l'utilisateur ait bien entré la forme de sa séquence d'où le choix du type booléen (true, false).

Pour la bonne gestion de cette fonction, l'utilisation des conditions : if, elif et else, était indispensable pour éviter les bugs dus à une mauvaise écriture et l'utilisateur sera toujours prévenu en cas d'erreur.

Dans les 3 conditions utilisées sur cette partie, nous avons également rajouter des sons « winsound » pour indiquer la réussite ou l'échec d'une entrée.

VI. Exécute : def execute() :

Cette fonction exécutera le programme. Elle va premièrement valider toutes les données que l'utilisateur a entrées au préalable avant exécution. En cas de valeur manquante, elle affichera les champs manquants et l'utilisateur pourra les remplir. De ce fait, l'utilisateur pourra utiliser les programmes étapes par étapes car il est impossible d'exécuter des données qui ne sont pas initialisées au départ.

Pour cette fonction, nous avons choisi d'utiliser des variables « ide, idf et ids » de type booléen pour valider ou non une entrée ; elles permettent également de rappeler à l'utilisateur les enzymes choisis ainsi que leur site de restriction.

Elle affichera ensuite un message disant à l'utilisateur que tout est bien été initialisé et l'invitera à lancer le traitement des données en appuyant sur « enter ». Le programme sera ensuite exécuté et rappellera à l'utilisateur les enzymes qu'il a choisit ainsi que leur site de restriction.

Cette fonction comprend trois fonctions en parallèle :

- Digestion d'une séquence linéaire : def digestDNAlinéaire (sequence, rs) et Digestion d'une séquence circulaire : def digestDNAcirculaire (sequence, rs) :

  Nous avons initialisé trois variable « 'frags', 'last' et 'g' » dans lesquelles sont stockés respectivement les fragments coupés, la position du dernier fragment enregistré (et donc coupé) et la position de coupure dans un site de restriction.

  Pour les deux digestions, nous avons utilisés les mêmes variables mais pour la digestion d'une séquence circulaire, nous avons ajouté une variable globale « flast » pour différencier avec « last » utilisée dans la digestion linéaire car la digestion d'une séquence linéaire ne donne pas le même nombre de fragments qu'une digestion circulaire. La variable globale « flast » va stocker le dernier fragment coupé afin de plus tard le coller au premier fragment coupé comme si la séquence était circulaire.

  Nous avons utilisé while 1 car notre boucle est longue et étroite, il est plus rapide en performance que while true.
  Nous avons utilisé les condition « if, elif et else » car la digestion devrait être faite soit pour une seule enzyme, soit deux, soit trois. Nous avons donc réfléchi en ordre de coupure pour les sites de restrictions en fonction du nombre d'enzymes utilisés (trois au maximum).

  Pour la digestion à deux et trois enzymes, nous avons utilisés des variables « i1, i2 et i3 en plus pour les trois enzymes) pour positionner les sites de coupure en fonction de l'ordre des enzymes utiliser : pour une seule enzyme, nous n'avons utilisé que la variable « i », pour deux enzymes « i1 pour la première enzyme » et « i2 pour la deuxième enzyme » et pour trois enzymes « i3 en plus pour le troisième enzyme ».

  Dans les conditions utilisées : pour le choix de deux enzymes et trois, lors du positionnement du sites de coupure, nous avons affecté « 10000000000000000 » à la variable i1, i2 ou i3 pour faciliter la comparaison des sites de coupure entre eux : nous avons donc réfléchi par ordre d'apparition du site.

- Migration (par ordre croissant + affichage électrophorèse) : def migrationfrag(taille) :

  Nous avons défini des variables globales « id1 à id20 » ainsi que « electro » afin de pouvoir les utiliser dans toutes la fonction. Elles serviront au positionnement des traits pour l'électrophorèse. Pour cela, nous avons réfléchi en intervalle (poids moléculaire) pour situer les différentes paires de base en bande d'électrophorèse.

Nous avons également utilisé les conditions « if, for, elif et else » pour afficher la migration par ordre croissant de longueur avec les bandes d'électrophorèse en fonction des différents intervalles de paires de base pour les deux séquences : linéaire et circulaire.

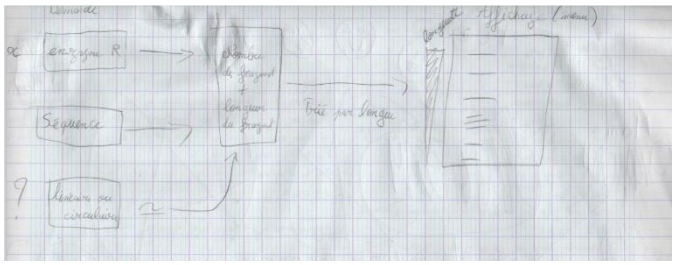
Pour faciliter le retour au menu, une fonction « def back() » a été mise en place ; et une autre fonction « def exit() » pour arrêter le programme.

MBANI-TALAMESSO Lise
MADIKITA BOKATOLA Divine
PRAGASSAM Anthony
CHERIF Aimen

L2-SDV-G4
ISV43

Carnet de bord

-[27 Février 20] : Nous nous sommes mis en quadri-nôme et nous avons décidé de choisir le sujet 4 du DM. Puis chacun a commencé à faire des recherches sur ce sujet de son côté et nous les



Première ébauche du programme

communiquant via WhatsApp dans le groupe que nous avons créé pour l'occasion.

-[23 mars 20] : L'un des premiers problèmes rencontrés a été la mise en place de menu. Après recherche, nous avions compris qu'il fallait utiliser des fonctions pour naviguer dans un menu. Pour nous aider dans la réalisation des fonctions que nous maitrisions mal, nous nous sommes aidés d'internet qui nous a apporté un squelette de menu que nous avons utilisé comme base. Au début nous avions compris comment l'utiliser et ajouter des fonctions au menu mais sans exactement comprendre comment il fonctionnait. Ce n'est qu'en faisant des tests et en avançant dans le programme (quelques semaines plus tard) que nous en avons compris totalement tout ce que faisait la fonction et qui nous a finalement permis d'utiliser avec aisance les fonctions dans le menu d'exécution (4).

-[24 mars 20] : Mise en place des fonctions d'entrée de la séquence et de la forme avec des vérifications lors de l'entrée des données respectives. La vérification de la séquence à poser un problème car ne fonctionnait pas avec les moyens habituels (chaine de caractère) pour des raisons que nous ignorons toujours. Quelques semaines plus tard, nous avons réglé ce problème en utilisant un dictionnaire pour vérifier la présence des lettres A, T, G et C.

-[25 Mars 20] : Nous avons rassemblé dans un fichier texte les enzymes de restriction ainsi que leur site de coupure récolté sur des sites internet comme Enzyme Finder, NEB. On a décidé de travailler qu'avec des enzymes de restriction de type II car c'est ceux que nous connaissons le mieux d'après nos cours de génie génétique et pour éviter également des bugs dans le programme.

-[27 mars 20] : Mise en place de la fonction d'entrée de l'enzyme par l'intermédiaire de fichier texte sur le modèle de l'exemple vu en cours.

-[28-30 Mars 20] : Recherche des informations pour programmer les vérifications des informations entrées par l'utilisateur dans les premières versions du programme (par exemple vérifier si la séquence d'ADN entrer est bien une séquence d'ADN)

-[30 mars 20] : Mise en place de vérification pour empêcher qu'une entrée soit manquante ou une entrée des données erronées et surtout que ces données erronées ou manquante soit envoyés pour l'exécution. Cela a été utile pour la suite en prévenant des bugs, surtout la double vérification avant l'exécution qui nous a permis de découvrir des erreurs de programmation lorsque les deux

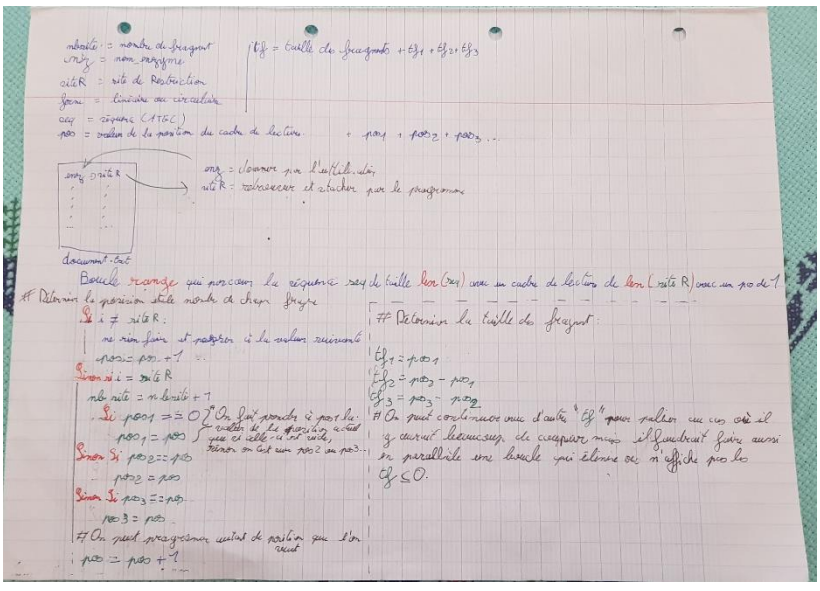
vérifications ne donnaient pas le même résultat (l'un nous disait que l'entrée des donnés était bon quand l'autre nous dit que l'entrée des donnés est invalide). D'autres vérifications ont été mise en place dans les autres fonctions pour améliorer celle faite la semaine plus tôt.

-[2 Avril 20] : Nous avons remarqué dans le programme un bug lorsqu'on entrait une longue séquence d'ADN, le programme bouclais sur le menu principal sans que l'on puisse rien faire.

-[3 avril 20] : Nous avons remarqué que le bug apparaissait lorsqu'on entrait une séquence d'ADN avec des sauts de ligne et que, le programme bouclait autant de fois qu'il y avait de saut de ligne dans la séquence d'ADN entrée. On a ensuite compris que lorsque l'on copiait/collait une séquence, celle-ci se copiait dans python avec des sauts de lignes. Ensuite python prenait chaque ligne pour une entité à part et chaque ligne était considérée comme input séparément. On avait donc la première ligne qui était enregistrée comme la séquence puis la seconde ligne était entrée comme choix du menu. Bien sur ce choix du menu était totalement invalide ce qui activait une protection qui faisait que si on entrait dans le choix du menu une entrée invalide, on était redirigé vers le menu principal. La troisième ligne était elle aussi comprise comme une entrée invalide ce qui activait à nouveau la protection et finalement faisait boucler le programme autant de fois qu'il y avait de ligne dans la séquence d'ADN entrée. Ne pouvant pas régler ce problème lié aux limites de python nous avons ajouté une protection qui stop complètement le programme en cas de trop nombreuse entrée invalide à la suite (la séquence étant de toute façon perdue et le programme pouvant boucler un long moment) par l'intermédiaire d'un compteur. Ce compteur compte chaque entrée invalide et se remet à 0 à chaque fois que l'on entre dans un menu pour limiter l'activation de cette protection qui était faite au départ pour des erreurs de choix du menu de l'utilisateur.

-[6 avril 20] : Réorganisation de tout le programme maintenant séparé en différentes catégories qui étaient mal structurées et donc difficilement compréhensibles lors de la relecture. La plupart des titres des catégories sont en anglais pour un rendu plus professionnel.

-[7 avril 20] : Lors du démarrage de nos essais pour la partie exécution du programme qui va traiter les données, nous avons été confrontés à un problème pour récupérer les données entrées par l'utilisateur d'une fonction à l'autre lié à notre manque d'expérience dans l'utilisation des fonctions. Comme exemple, une séquence entrée dans le menu séquence n'existait que dans ce dit menu et était effacée dès que l'on en sortait. En effet, nous n'avions jamais testé la présence d'une valeur en dehors de sa fonction d'entrée. Ce problème remettait en cause tout le programme car empêchait l'entrée des données. Ce n'est qu'après plusieurs heures de recherche que nous avons découvert la fonction « Global » qui permettait qu'une modification apportée à une valeur (donc l'entrée de données) dans une fonction soit valable dans tout le programme et plus seulement dans une seule fonction.



Première ébauche de la partie exécution.

-[6-11 avril 20] : Mise en place des premières ébauches du programme de traitement données grâce à une base trouver sur internet que nous avons amélioré et adapté pour qu'elle fonctionne dans notre programme et par la suite puisse supporter l'entrée de plusieurs enzymes. Le problème a été de comprendre comment fonctionnait la base trouvée sur internet pour pouvoir le modifier à notre convenance. Nous avons justement découvert un bug qui empêchait l'entrée d'une enzyme qui coupe en position 0 de son site de restriction et qui était là aussi lié aux limites de python. Nous avons donc choisi de ne pas exactement respecter la réalité biologique et à faire en sorte que la coupure soit en position 1 dans ce type de cas. Une fois le fonctionnement du programme maitrisé nous avons intégré le fait de pouvoir entré plusieurs enzymes de restriction. Nous avons choisi de faire en sorte que les enzymes et leurs sites ne soit plus stocker dans une simple chaine de caractère mais dans une liste car plus simple à manipuler. Comme nous avions compris comment bien utiliser le programme de traitement des données, il n'a pas été difficile de faire ces modifications. Nous avons tout de même limité le nombre d'enzyme à trois maximums car le code de traitements des données devenait trop compliqué (Edit : Quelques semaines plus tard, nous avons pu simplifier le code qui devenait reproduisible pour plus de 3 enzymes mais nous avons choisi de ne pas nous éparpiller et de nous limiter à 3 enzymes maximums). Un compteur va compter le nombre d'enzymes, empêcher qu'on en ajoute plus que le maximum et permettre de traiter les données différemment selon le nombre de cas. Un dernier petit problème dans le menu de choix des enzymes a été le nommage de toutes les valeurs pour que cela reste logique et compréhensible.

-[14 avril 20] : Le programme devenait de plus en plus chargé et il nous était de plus en plus long et difficile de comprendre les codes écrits par nos coéquipiers donc nous avons organisé une visioconférence par Skype pour expliquer à tout le monde les fonctionnalités du programme (pour parler du code, la résolution des bugs et l'organisation du reste du travail qui restait à faire) et régler ce problème de compréhension au sein du groupe.

-[20 avril 20] : Visioconférence avec Mme Diaz. Nous découvrons que nous avions eu un petit problème de compréhension du sujet. En effet nous avions fait un affichage des fragments dans l'ordre de coupure puis simulé une migration sur gel avec ces fragments alors qu'il nous était seulement demandé de faire un affichage de fragment dans l'ordre croissant. Nous avons donc choisi de garder la simulation de migration sur gel mais de modifier l'affichage des fragments pour qu'ils soient affichés de façon ordonnée croissante.

-[11 Mai 20] : Rédaction de l'algorithme du programme en Français.

-[12 Mai 20] : Rédaction du mode d'emploi du programme : explications détaillées.

-[20 mai 20] : Mise en place de la partie du programme faisant l'affichage des fragments obtenus après digestion triés par ordre croissant. Nous avons donc choisi de passer par les listes car on s'est aperçu que c'était plus simple en utilisant les méthodes. On a donc initialisé une liste vide pour chaque forme d'ADN (linéaire ou circulaire) avant de lui rajouter les fragments obtenus par digestion avec la méthode « append() » et de trier ces derniers par ordre croissant selon leurs longueurs en pb à l'aide d'une deuxième méthode « sort() »

-[21 mai 20] : Des dernières vérifications et discussions ont été faites par l'ensemble du groupe notamment pour d'éventuelles améliorations principalement visuelles.

-**[22 mai 20]** : Découverte du fait qu'en cas de séquence circulaire, il est possible qu'il y ait apparition d'un nouveau site de restriction aux jonctions de la séquence. Au vu de la découverte tardive de ce problème par rapport à la date de rendu de travail, de la refonte du programme que cela

impliquerait de le régler et de la faible possibilité que cela puisse se produire, nous avons choisi de ne pas respecter la réalité biologique et d'ignorer ce petit problème.

-[15-23 Mai 20] : Nous avons rassemblé tous les liens des sites que nous avons utilisé pour nos recherches pour le DM et nous avons fait notre bibliographie.

-[18-23 mai 20] : Rassemblement de nos journaux de bord pour préparer cette fiche

MBANI-TALAMESSO Lise
MADIKITA BOKATOLA Divine
PRAGASSAM Anthony
CHERIF Aimen

L2-SDV-G4
ISV43

Bibliographie

Sites internet

- *REBsites* [en ligne]. Disponible sur : http://tools.neb.com/REBsites/index.php. Consulté le 10 mars 2020.
- Dr. Richard J. Roberts et Dana Macelis. *Rebase* [en ligne]. D298-D299 (2015). Disponible sur : http://rebase.neb.com/rebase/rebase.files.html. Consulté le 14 mars 2020.
- *Sequence Manipulation Suite (SMS)* . [en ligne]. Mise à jour le 6 Novembre 2017. Disponible sur : https://www.bioinformatics.org/sms2/rest_digest.html. Consulté le 27 février 2020.
- *Bg go further*. [en ligne]. Mise à jour le 12 janvier 2015. Disponible sur : https://www.bggofurther.com/2015/01/create-an-interactive-command-line-menuusingpython/. Consulté le 5 mars 2020.
- *Wikibooks*. [en ligne]. Mise à jour le 16 avril 2020. Disponible sur : https://fr.wikibooks.org/wiki/Programmation_Python/Fonctions. Consulté le 25 mars 2020.
- Vincent le Goff. *Apprendre à programmer en python* in OPENCLASSROOMS. [en ligne] Mise à jour le 13 mai 2020. Disponible sur : https://openclassrooms.com/fr/courses/235344apprenez-a-programmer-en-python/231174creez-des-structures-conditionnelles. Consulté le 5 Avril 2020
- *GammigHacks*. *Python print en couleur* .[en ligne]. Disponible sur : http://gaminghacks.free.fr/Python,%20print%20en%20couleur.php. Consulté le 5 Avril 2020.
- *Enzyme Finder*. [en ligne]. 2017-2018. Disponible sur : https://enzymefinder.neb.com/#!/overhang/?category=all&type=any&exact=yes&exactlen=yes%23nebheader#nebheader. Consulté le 10 mars 2020.
- JPSoftware. *Beep* . [en ligne]. Disponible sur : https://jpsoft.com/help/beep.htm . Consulté le 16 mars 2020.
- Cours de python. *Plus sur les chaines de caractères*. [en ligne]. Disponible sur : https://www.digitalbiologist.com/blog/2011/04/code-tutorial-getting-startedwithpython.html. Consulté le 22 février 2020.
- The digital biologist. *Code Tutorial: Getting started with Python in the lab* [en ligne]. Mise à jour le 3 Avril 2011. Disponible sur : https://www.digitalbiologist.com/blog/2011/04/codetutorial-getting-started-with-python.html. Consulté le 11 mars 2020.
- Mohamed Ouamer. Programmation 360. *Listes en Python: Création, Modification et Suppression.* [en ligne]. Disponible sur : https://programmation360.com/listes-python/. Consulté le 04 avril 2020.
- Python Software Foundation. *D'autres outils de contrôle de flux.*[en ligne]. Mise à jour le 15/02/2020. Disponible sur : https://docs.python.org/fr/3/tutorial/controlflow.html. Consulté le 15 mars 2020.

- Matthieu Schaller & les validateurs. Openclassrooms. *Utilisation avancée des listes en Python*. [en ligne]. Mis à jour le 31/10/2013. Disponible sur : https://openclassrooms.com/fr/courses/1206331-utilisation-avancee-des-listes-en-python. Consulté le 18 mars 2020.
- Pdave. it-swarm.dev. *Comment arrêter une boucle infinie en toute sécurité en Python?*. [en ligne]. Mise à jour le 03/10/2015. Disponible sur : https://www.itswarm.dev/fr/python/comment-arreter-une-boucle-infinie-en-toute-securiteen-python/1055067409/. Consulté le 11 avril 2020.
- Cours de python. *Les variables*. [en ligne]. Disponible sur : https://python.sdv.univ-parisdiderot.fr/02_variables/ . Consulté le 20 mars 2020.
- Très Facile. *Python Très Facile*. [en ligne]. Disponible sur : https://www.tresfacile.net/tousles-cours/. Consulté le 8 avril 2020.
- David Cassagne. *Cours de python*. [en ligne]. Mise à jour le 15/12/2019. Disponible sur : https://courspython.com/dictionnaire.html. Consulté le 22 avril 2020.
- Openclassrooms forum. *[Python] Remplacer une chaine de caractère par une autre*. [en ligne]. Disponible sur : https://openclassrooms.com/forum/sujet/python-remplacerunechaine-de-caractere-par-une-autre-48823. Consulté le 25 avril 2020.
- Developpez.net forum. *[Python] Trier une liste de string selon la longueur de chaque élément*. [en ligne]. Disponible sur : https://www.developpez.net/forums/d1538445/autreslangages/python/general-python/trier-liste-string-selon-longueur-elements/

Vidéos en ligne

- Programmers' Colloquy. *Python Tutorial For Beginners | Add Sound In Python Program | Winsound*. [vidéo en ligne]. Youtube, mise en ligne le 06/02/2019. Dispinible sur : https://www.youtube.com/watch?v=9FOoYbF42h0 . Consulté le 3 mars 2020. 1 vidéo, 14 min.
- Formation Vidéo. *Python #19 – fichiers*. [vidéo en ligne]. Youtube, mise en ligne le 30/08/2017. Disponible sur : https://www.youtube.com/watch?v=gVOYPwjd_8c. Consulté le 3 mars 2020. 1 vidéo, 30min.
- Graven - Développement . *APPRENDRE LE PYTHON #7 ? LES OBJETS*. [video en ligne]. Youtube, mise en ligne le 24/11/2018. Disponible sur : https://www.youtube.com/watch?v=dfUM_9xibf8. Consulté le 18 mars 2020. 1 video, 20 min.
- Formation Vidéo. *Python #9 - gestion erreurs*. [vidéo en ligne]. Youtube, mise en ligne le 16/03/2017. Disponible sur : https://www.youtube.com/watch?v=1IqnpaQy8LM&list=PLrSOXFDHBtfHg8fWBd7sKPxEmahwyVBkC&index=9 . Consulté le 5 mai 2020. 1 vidéo, 33 min.
- Formation Vidéo. *Python #18 – dictionnaires*. [vidéo en ligne]. Youtube, mise en ligne le 05/07/2017. Disponible sur : https://www.youtube.com/watch?v=BYDJRsE-N5Y. Consulté le 29/04/2020. 1 vidéo, 31 min.

- Graven – Développement. *APPRENDRE LE PYTHON #10 ? LES FICHIERS*. [video en ligne]. Youtube, mise en ligne le 19/09/2019. Disponible sur : https://www.youtube.com/watch?v=jOHpZg8k668&t=7s. Consulté le 18/04/20. 1 vidéo, 18 min.