Algorithmique

License 1 - Pôle 3D

Vincent Reynaert

Les objectifs

Objectifs Communs :

- Comprendre la notion d'algorithme
- Savoir décrire/écrire un algorithme en python, en pseudo langage (en langage humain) ou sous forme de diagramme
- Comprendre les bases des langages de programmation et être capable d'implémenter des algorithmes sous Python pour programmer un jeu simple en mode texte

Objectifs Avancés :

- Découvrir l'interface graphique sous Python
- Réfléchir à des problèmes algorithmiques plus poussés
- Être capable de faire un jeu avec une logique algorithmique avancée et\ou avec une interface graphique

Les évaluations

Evaluation individuelle (format interro écrite) :

- QCM sur des connaissances “théorique”
- Comprendre ce que fait tel ou tel algorithme écrit en Python ou en pseudo langage
- Proposer un algorithme en pseudo langage pour répondre à un problème donné

Evaluation par grp (bi-trinômes) :

- Proposer un jeu simple en mode texte mettant en place des connaissances algorithmiques vue en cours ou découverte par vous même
- Bonus : proposer une interface graphique ou mettre en place des algorithmes plus poussés

Programme

Chap 1 : Algorithme (recette de cuisine)

Chap 2 : Variables et Types de Données

Chap 3 : Conditions (Si, Sinon Si, Sinon)

Chap 4 : Boucles (Pour, Tant que)

Chap 5 : Fonctions (f(x)->y)

Chap 6 : Python et Jupyter (installation et base de syntaxe)

Chap 7 : Implémentation (traduire nos algo papiers en algo informatique)

Chap 8 : Listes / Tableaux

Chap 9 : Turtle

Chap 1 : Algorithme (recette de cuisine)

Trouver le nombre entre 0 et 2048

https://images.math.cnrs.fr/L-informatique-sans-ordinateur

Je choisi un nombre en 0 et 2048.

Votre objectif est de trouver ce nombre le plus vite possible (en faisant le moins de proposition possible)

Je ne répondrais qu'en indiquant si le nombre que j'ai choisi est strictement inférieur ou supérieur à celui que vous me proposez ou s'il est égal auquel cas vous avez gagné

Trouver le nombre entre 0 et 2048

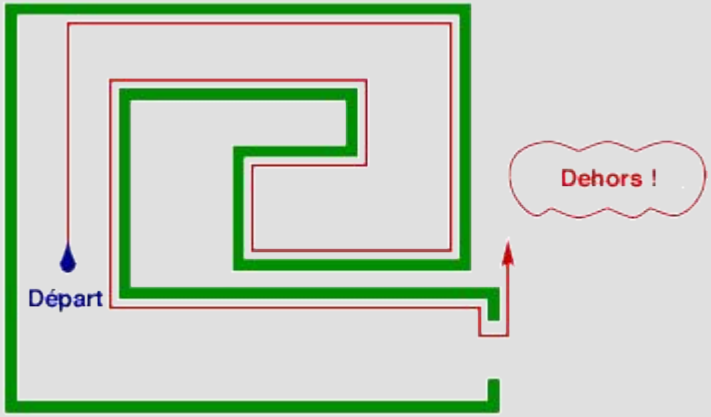
Pour trouver le plus rapidement possible le résultat, il faut appliquer un principe de recherche Dichotomique :

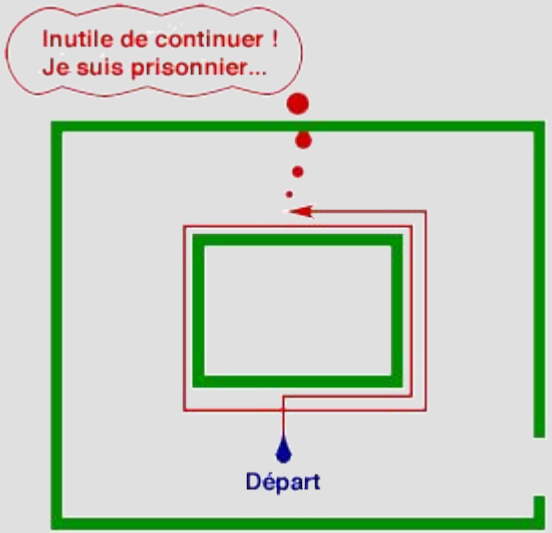
- C'est à dire que nous prenons l'intervalle dans lequel se trouve le nombre que l'on cherche et nous le coupons en deux parts égales.
- Si le nombre proposé est le bon c'est gagné
- Sinon si le nombre est plus grand ou plus petit que celui recherché, nous prenons le nouvel intervalle formé par le nombre proposé et et on renouvelle l'opération

Trouver la sortie d'un Labyrinth

https://images.math.cnrs.fr/Dis-maman-ou-papa-c-est-quoi-un-algorithme-dans-ce-monde-numerique

https://interstices.info/lalgorithme-de-pledge/





S’entraîner progressivement

- Débuter : https://blockly.games/maze?lang=fr&level=7&skin=0
- Aller plus loin (attention peut-être assez compliqué) : https://www.codingame.com/start

Définition d’un Algorithme

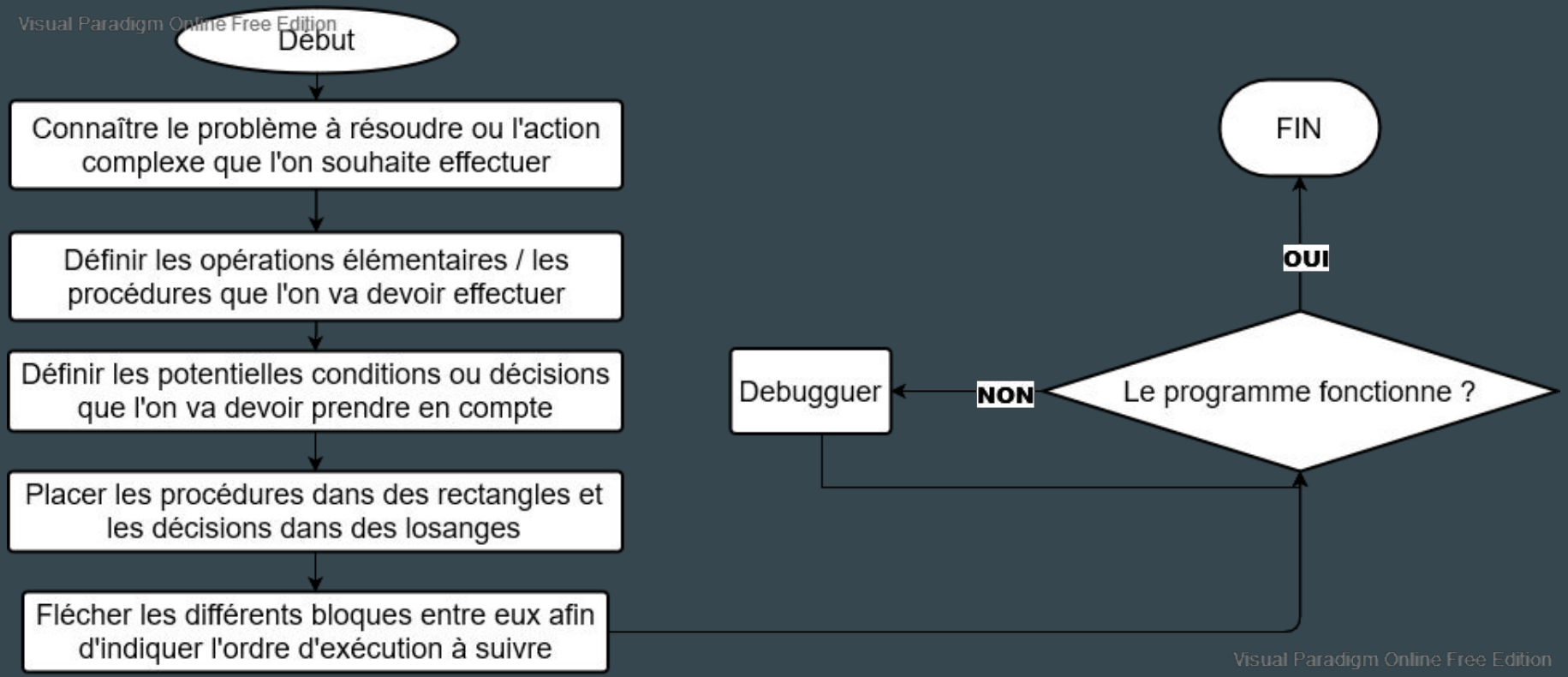
Un **algorithme** est une suite finie et non ambiguë d'instructions et d’opérations permettant de résoudre certains problèmes ou d’effectuer certaines actions.

Il a donc un début et une fin entre lesquels nous trouvons des procédures et des décisions qui se suivent.

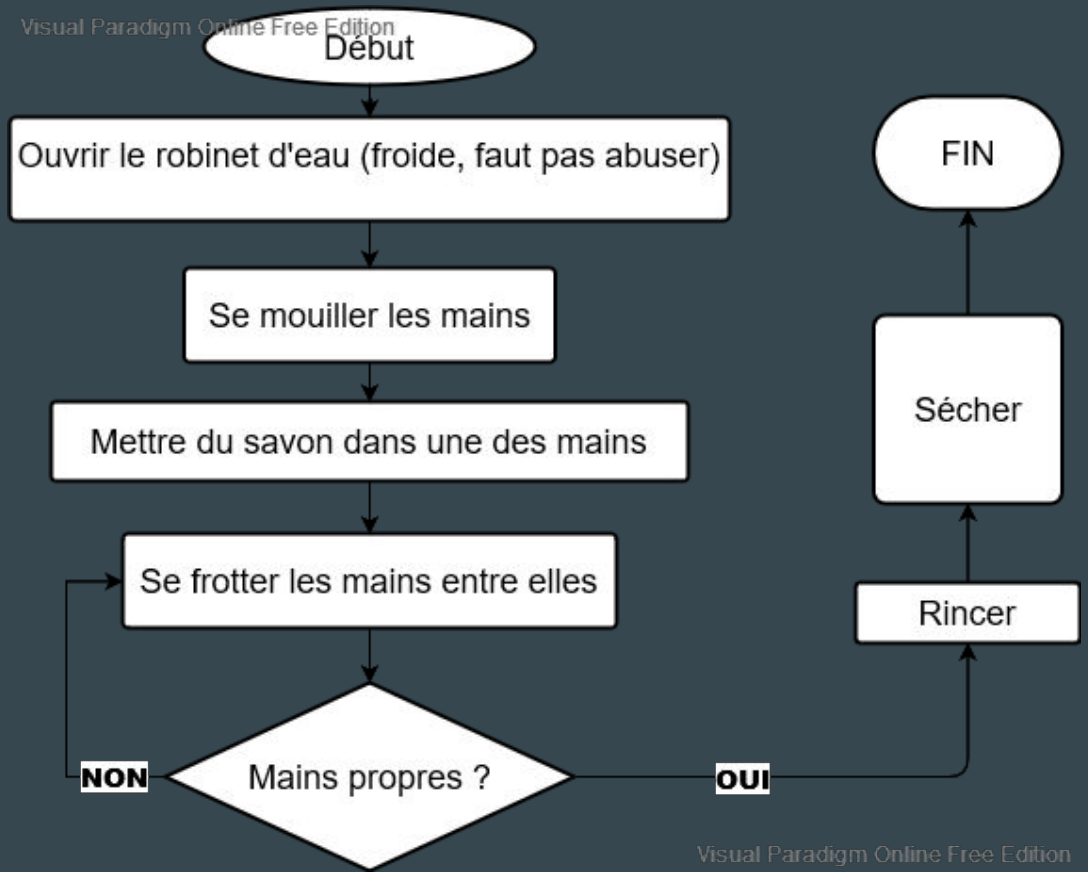
Exemples :

- recette de cuisine
- notice de montage
- itinéraire
- programme informatique
- etc.

Représentation graphique d'un Algorithme



Exemple : L'algo du lavage de mains



Trouver le nombre entre 0 et 2048

Exemple :

Quelqu'un veut me faire trouver le nombre 500.

Je vais donc proposer 2048/2 - > 1024. Il va m'annoncer que son nombre est plus petit

Ensuite :

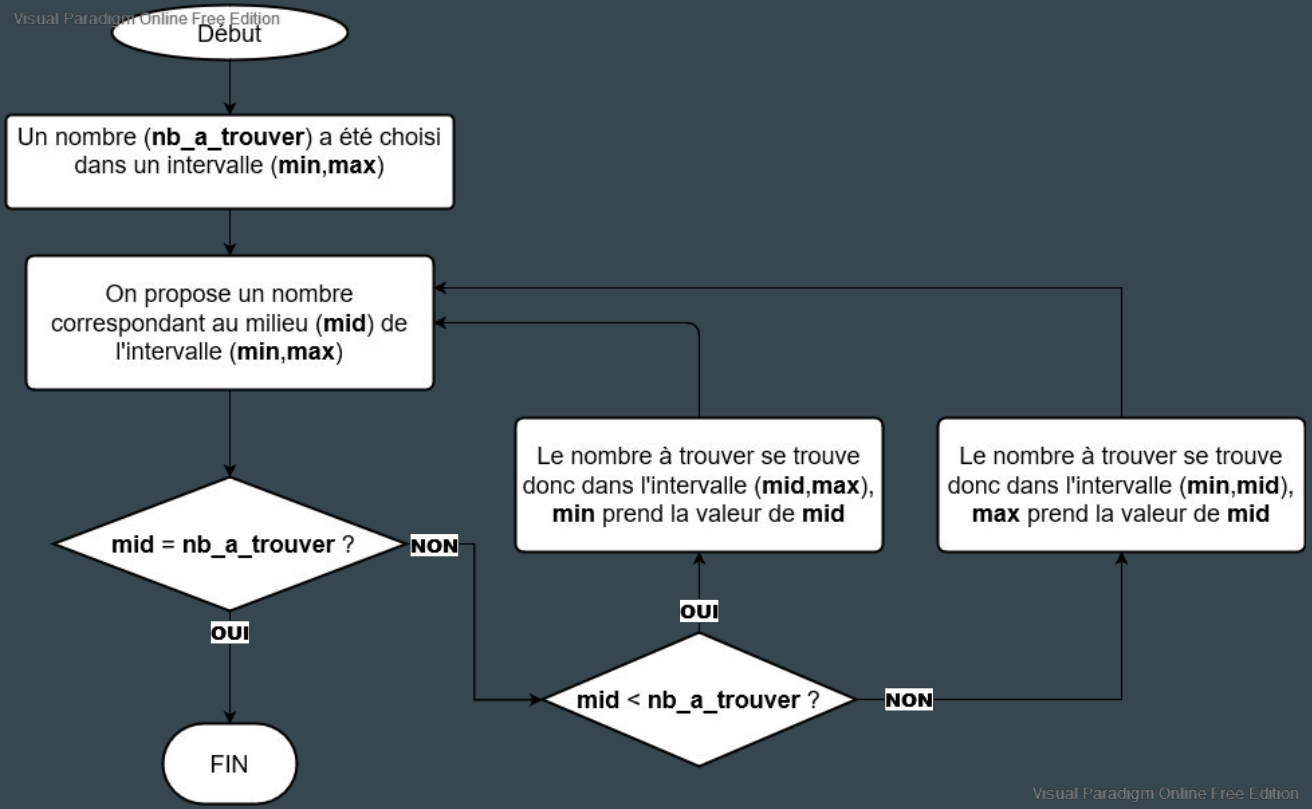
Je propose	Il répond	Je propose	Il répond
512	son nombre est plus petit	480	son nombre est plus grand
256	son nombre est plus grand	496	son nombre est plus grand
384	son nombre est plus grand	504	son nombre est plus petit
448	son nombre est plus grand	500	BRAVO !

Trouver le nombre entre 0 et 2048

Pour trouver le plus rapidement possible le résultat, il faut appliquer un principe de recherche Dichotomique :

- C'est à dire que nous prenons l'intervalle dans lequel se trouve le nombre que l'on cherche et nous le coupons en deux parts égales.
- Si le nombre proposé est le bon c'est gagné
- Sinon si le nombre est plus grand ou plus petit que celui recherché, nous prenons le nouvel intervalle formé par le nombre proposé et et on renouvelle l'opération

Trouver le nombre entre 0 et 2048



Chap 2 : Variables et Types de Données

Ressources utiles

Pour débuter :

https://youtube.com/playlist?list=PLG38RdG_6oUNLtTJfWGz7t11dhP6aTkWv

Pour aller un peu plus dans le détail :

https://fr.wikipedia.org/wiki/Variable_(informatique)

Et encore plus loin :

https://algo.developpez.com/tutoriels/initiation/

Définition de Variable

Une **variable** est une zone de la mémoire de l'ordinateur dans laquelle une **valeur** est stockée. Aux yeux du programmeur, cette variable est définie par un **nom**, alors que pour l'ordinateur, il s'agit en fait d'une adresse, c'est-à-dire d'une zone particulière de la mémoire.

Autrement présentée : une variable peut être considérée comme une boîte que l'on va étiqueter pour lui donner un nom et dans laquelle on va stocker une information. Par exemple, le nom d'un utilisateur, son âge ou la vie d'un personnage de jeu vidéo son stocké dans des variables.

L’utilité des Variables

Les variables peuvent être appelées par le nom qu’on leur aura donné (l’étiquette du carton) pour récupérer et afficher leur contenu ou faire des opérations avec ou encore modifier ce contenu.

Par exemple : la vie d’un joueur dans le jeu, est amenée à évoluer au cours de la partie. Si le joueur perd de la vie on appellera la variable ***Vie*** que l’on aura préalablement créée et on va la faire diminuer avant de remettre la nouvelle valeur dans la variable ***Vie*** (on parle alors de **décrémenter** la variable). L’opération inverse, soit l’ajout de point de vie fonctionnera de la même façon et sera appelée **incrémentation**

Création de Variables

La création de la variable (c'est à dire créer la boîte et mettre l'étiquette) s'appelle la **déclaration**.

Ensuite, donner une valeur à cette variable (stocker un élément dans la boîte) correspond à l'**affectation**.

En pseudo code on pourrait écrire comme suit :

- Déclarer la variable pointsDeVie
- Affecter la valeur 3 à la variable pointsDeVie

Pour faire plus simple on pourra également écrire :

- variable pointsDeVie
- 3 → pointsDeVie

Il arrive même parfois que dans certains langages (comme le Python) la déclaration soit implicite et puisse se faire en même temps que l'on affecte la première valeur à notre variable :

- 3 → pointsDeVie

Règles sur les noms des Variables

- Le nom d'une variable est unique, on ne peut avoir deux variables du même nom dans un même programme.
- Le nom d'une variable ne peut être composée uniquement de chiffre mais elle peut en contenir (ex: 1234 n'est pas un nom approprié mais Fantasio974 est un nom approprié).
- Le nom d'une variable ne peut contenir de caractères spéciaux autres que _ (underscore ou tiret du 8) (ex: *points de vie* n'est pas un nom approprié mais *pointsDeVie* ou *points_de_vie* sont des noms appropriés).
- Une variable doit avoir un nom explicite (à savoir que des abréviations communéments utilisées sont acceptables mais si vous voulez introduire vos propres abréviations alors il faudra au moins indiquer en commentaire à quoi elle correspond plus précisément).
    - Choisissez au début de votre programme si vous allez écrire les noms des variables en fra ou eng pour une question de cohérence

Type de données

Les variables peuvent contenir différents types de données :

- numérique comme les points de vie ou l'âge qui sont des nombres (entier ou à virgule, positif ou négatif)
- alphanumérique comme un prénom ou un dialogue qui sont des chaînes de caractères généralement délimitées par des “” (ex : “Un super exemple”, “et un 2nd”)
- booléen qui est une donnée qui ne peut avoir que deux valeurs : Vrai ou Faux (True or False, 1 ou 0)
- les dates sont des types de données plus complexes tout comme les couleurs

Définir quel type de données une variable devra stocker va permettre à l'ordinateur de savoir quelle taille de boîte il faudra prévoir (ex : si on veut mettre un pc portable ou un frigo dans un carton, on n'aura pas besoin de la même taille d'emballage).

Cependant certain langage comme le python ne demande pas à définir le type lors de la déclaration de la variable alors que le C# (langage de programmation pour Unity) le demande.

Les opérations

Sur des données numériques les opérations mathématiques de base sont tout à fait possibles (ex: addition 1+2 = 3, soustraction 9-4 = 5, multiplication 5*8 = 40, division 6/4 = 1.5, division entière 6//4 = 1, modulo (reste de la division entière) 6%4 = 2, puissance 2**4 ou 2^4 ou pow(2,4) = 16).

Sur les chaînes de caractères les opérations possibles dépendent du langage mais basiquement nous retrouverons au moins la *concaténation* qui consiste à coller des chaînes de caractères entre elles (ex: “A ” + “2 mains ” + “ou pas” = “A 2 mains ou pas”). (attention on ne peut pas faire “A ” + 2 + “ mains” car 2 n’est pas une chaîne de caractère s’il est en dehors des “”)

Sur les booléens certaines opérations logiques sont possible (ex: (pas Vraie) = Faux, (Vraie ou Faux) = Vraie, (Vraie et Faux) = Faux).

Chap 3 : Conditions (Si, Sinon Si, Sinon)

Exemple : Si / If

On veut notifier au joueur que l'ennemi est mort si ce dernier n'a plus de points de vie :

```
pv_ennemi ← 4 # ce qui est écrit ici en vert après le dièse ou hashtag est un commentaire qui donne de l'information complémentaire au lecteur de l'algorithme mais qui n'est pas pris en compte pour le résoudre, pour l'ordinateur c'est comme si les commentaires n'existaient pas
point_de_degats_joueur ← 3
pv_ennemi ← (pv_ennemi - point_de_degats_joueur)
Si pv_ennemi = 0 alors # ici nous observons que pv_ennemi est différent de 0, le booléen (pv_ennemi = 0) est FAUX alors on n'entre pas dans la condition
        Afficher "L'ennemi est mort"
Fin Si
pv_ennemi ← (pv_ennemi - point_de_degats_joueur)
Si pv_ennemi = 0 alors # ici aussi on observe que pv_ennemi est différent de 0, (pv_ennemi vaut -2) Il aurait fallu tester Si pv_ennemi < 0
        Afficher "L'ennemi est mort"
Fin Si
```

Exemple : Si , Sinon / If , Else

On veut notifier au joueur que l'ennemi est mort si ce dernier n'a plus de points de vie sinon on l'attaque à nouveau:

```
pv_ennemi ← 4
point_de_degats_joueur ← 3
pv_ennemi ← (pv_ennemi - point_de_degats_joueur)
Si pv_ennemi < 0 alors # ici nous observons que pv_ennemi est différent de 0, le booléen (pv_ennemi < 0) est FAUX alors on passe au SINON
    Afficher "L'ennemi est mort"
Sinon
    pv_ennemi ← (pv_ennemi - point_de_degats_joueur)
Fin Si
```

Exemple : Si , Sinon Si, Sinon/ If ,Elif-Else If, Else

On veut faire bouger un joueur en fonction des touches appuyées du clavier ZQSD :

```
Si joueur appuie sur une touche alors
    touche_appuyee ← entrée clavier du joueur
    Si touche_appuyee = “Z” alors
        Déplacer joueur vers le haut
    Sinon Si touche_appuyee = “S” alors
        Déplacer joueur vers le bas
    Sinon Si touche_appuyee = “Q” alors
        Déplacer joueur vers la droite
    Sinon Si touche_appuyee = “D” alors
        Déplacer joueur vers la gauche
    Sinon
        Afficher “Veuillez appuyer sur une des touches suivantes : ZQSD”
    Fin Si
Fin Si
```

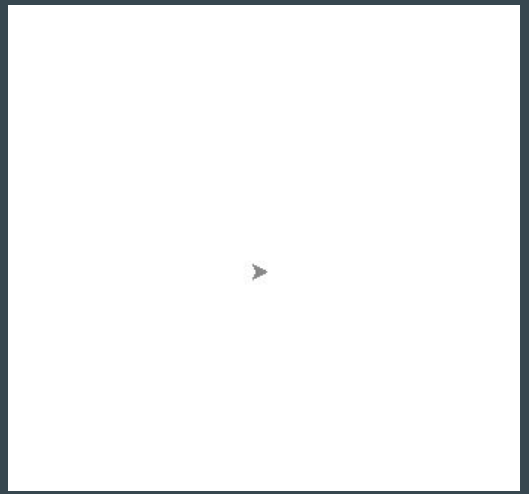
Chap 4 : Boucles (Pour, Tant que)

Les boucles

Les boucles permettent de répéter des instructions un certain nombre de fois ou jusqu'à ce qu'une certaine condition soit vérifiée.

Ex : Vous voulez afficher une rosace à l'écran et pour cela il vous faut dessiner un cercle puis un autre, puis encore un autre, toujours avec un léger décalage. Imaginons qu'on veuille faire une rosace à trois cercles :

Poser le crayon à la position (0,0)
Dessiner cercle de rayon 8
Tourner vers la gauche sur 360/3 degrés
Dessiner cercle de rayon 8
Tourner vers la gauche sur 360/3 degrés
Dessiner cercle de rayon 8



Les boucles

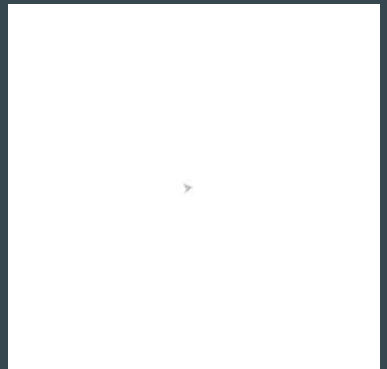
Ex : Pour trois cercles le code à écrire n'est pas trop long mais imaginons qu'on veuille en faire 30...

Poser le crayon à la position (0,0)
Dessiner cercle de rayon 8
Tourner vers la gauche sur 360/30 degrés
Dessiner cercle de rayon 8
Tourner vers la gauche sur 360/30 degrés
Dessiner cercle de rayon 8
Tourner vers la gauche sur 360/30 degrés
Dessiner cercle de rayon 8
...

Les boucles FOR / POUR

Ex : C'est la qu'on va pouvoir introduire la notion de boucle. Ici nous utiliserons la boucle FOR / POUR. Cela va nous permettre d'écrire sur 5 lignes un code qui en aurait pris près de 150 lignes sinon :

Poser le crayon à la position (0,0)
POUR compteur allant de 0 à 30 (avec un pas de 1) FAIRE # attention le 30 est généralement considéré comme exclu de la boucle, c'est à dire que le compteur prendre les valeurs 0,1,2,3...,28,29 mais pas 30, aussi le pas est souvent par défaut à 1 il n'est donc pas nécessaire de le préciser dans ce cas. Comme pour le SI, il faudra penser à bien indenter (tabuler) votre code
    Dessiner cercle de rayon 8
    Tourner vers la gauche sur 360/30 degrés
Fin POUR



Les boucles WHILE / TANT QUE

Ex : J'ai un ennemi qui a un certain nombre de points de vie et j'ai un personnage jouable qui possède un certain nombre de dégât, je veux tapper l'ennemi jusqu'à ce qu'il n'ai plus de points de vie. Je pourrais faire comme je l'ai montré dans les if, et écrire une suite d'instructions identiques mais ce n'est clairement pas optimisé et on pourrait en oublié. Posons que l'ennemi à 500 PV et que notre perso produit 3 points de dommages, il faudrait beaucoup trop d'instruction... on pourrait utiliser une boucle POUR si on calcule par avance que 500/3 cela demande de tapper 167x pour tuer l'ennemi mais ici nous le TANT QUE.

```
pv_ennemi ← 500
point_de_degats_joueur ← 3
TANT QUE pv_ennemi > 0 FAIRE
        pv_ennemi ← (pv_ennemi  - point_de_degats_joueur)
Fin TANT QUE
Afficher “L'ennemi est mort”
```

Les boucles WHILE / TANT QUE -> Boucle INFINIE ∞

ATTENTION : Les boucles WHILE sont dangereuse !

Ex : Si on se dit que pour qu'un ennemi meurt il faut que ses PV atteignent 0 on peut se dire que tant que ses PV ne sont pas égaux à 0 alors on continue de l'attaquer. Mais ici nous lui mettons 3 points de dégâts ce qui fait qu'au bout d'un moment nous allons arriver à pv_ennemi qui vaut 2 et si on lui soustrait encore 3 on passe à -1 ce qui est différent de 0... On se rend alors compte que jamais on ne sortira de la boucle

```
pv_ennemi ← 500
point_de_degats_joueur ← 3
TANT QUE pv_ennemi != 0 FAIRE # != signifie différent ou "non égal"
        pv_ennemi ← (pv_ennemi  - point_de_degats_joueur)
Fin TANT QUE
Afficher "L'ennemi est mort"
```

Chap 5 : Fonctions (f(x)->y)

Définition

Une fonction :

- Portion de code informatique nommée, qui accomplit une tâche spécifique.
- Généralement elles reçoivent des données en entrées et retournent le résultat du traitement de ces données.

Fonction Native / Build-in

Ces fonctions sont prédéfinies dans le langage et peuvent varier d'un langage à un autre.

En Python, nous avons déjà vu (ou nous verrons prochainement) :

- print(), qui en entrée prend un élément qu'il considère comme une chaîne de caractères et qui affiche cet élément (attention cette fonction ne retourne rien)
- eval(), qui en entrée prend une chaîne de caractères et retourne le contenu de la chaîne de caractère soit sous forme de chaîne de caractères (aucun changement s'il s'agit effectivement d'un texte), soit un nombre entier, soit un nombre à virgule, soit un autre type en fonction de ce qu'il y a dans la chaîne en entrée (il s'agit d'une fonction assez complète et complexe en sorte
- int(), qui en entrée prend un élément et si possible il le convertit en nombre entier
- input(), qui en entrée prend optionnellement un string qui est affiché et ensuite écoute ce qu' écrit l'utilisateur jusqu'à ce qu'il appuie sur la touche enter du clavier. Cette fonction retourne ce qu'a écrit l'utilisateur sous forme de string.

Fonction Personnalisée

Le programmeur écrit de nombreuses fonctions qu'il définit lui-même pour faire son programme.

Pourquoi écrire une fonction ?

- Décomposer un long programme complexe en plusieurs sous tâches plus simples et élémentaires.
- Une fonction peut être appelée à plusieurs reprise et permet donc d'éviter de copier coller des gros bout de code partout.

Définir une fonction

Voir le script jupyter pour plus d'exemple.

Une fonction python se définit comme suit (attention à l'incrémentation):

```
def nomDeMaFonction(param1,param2,etc.):
        #opération sur les param
        return leResultat #optionnel
```

La fonction la plus basique qu'il soit et qui ne fait rien s'écrira comme suit :

```
def fonctionDeBase():
        pass
```

Chap 6 : Python / Jupyter (installation et syntaxe)

Ressources utiles

Livre complet sur la programmation python adapté aux débutants/intermédiaires : https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&ved=2ahUKEwjyr8rsjq_zAhV68eAKHdSWDDUQFnoECBkQAQ&url=http%3A%2F%2Fwww.siteduzero.com%2Fuploads%2Ffr%2Fftp%2Flivre%2Fpython%2Fapprenez_a_programmer_en_python.pdf&usg=AOvVaw3hr7anytRd8cJCW29gijpa

L’algo d’installation de Python

- https://www.python.org/downloads/ (Télécharger Python 3.9.7)
- Lancer le fichier .exe téléchargé
- Cocher la case demandant d’ajouter Python au PATH / variables d’environnements
- Choisir l’installation personnalisée
- Vérifier que toutes les cases sont cochées sur la première page sinon les cocher
- Sur la page suivante ne rien modifier
- Finaliser l’installation
- Voilà

Lancer un fichier Python

- Lancer IDLE de python (taper python dans la barre de recherche de windows 10 devrait normalement faire apparaître IDLE comme premier programme trouvé)
- Un shell python (IDLE shell 3.9.7) apparaît, il s'agit de l'invite de commande Python. Il est compliqué d'y écrire des programmes complexes. Mais pour des commandes ne faisant qu'une ou deux lignes cela peut convenir.
- On peut par exemple taper :
  - *print("Hello World")*

  On verra alors apparaître la phrase **Hello World** dans l'interpréteur

- Dans l'onglet **File** créez un nouveau fichier (new file).
- Sauvegardez votre fichier sous le nom 'test.py' sur le bureau par exemple ou dans un dossier de travail.
- Une fois cela fait nous allons écrire notre premier programme Python.

Hello World

- tapez :
  - print(“Hello World”)

  dans votre fichier test.py

- ensuite dans l’onglet Run, cliquez sur Run Module. (vous pouvez également lancer le programme en appuyant sur le raccourci clavier F5)
- le programme se lance donc dans l’interpréteur (Shell) et affiche simplement Hello World comme lorsqu’on lançait la commande directement dedans.
- Ici nous utilisons la fonction print() pour afficher la phrase “Hello World”

Hello World bis

- effacez l'instruction précédente et tapez :
  - a = "Hello World"
  - print(a)

  dans votre fichier test.py

- ensuite dans l'onglet Run, cliquez sur Run Module. (vous pouvez également lancer le programme en appuyant sur le raccourci clavier F5)
- le programme se lance donc dans l'interpréteur (Shell) et affiche simplement Hello World comme lorsqu'on lançait le programme précédent.
- Ici nous avons Instancié la variable a et stocké le string / la chaîne de caractères "Hello World" dans cette variable.

Installation Jupyter

https://jupyter.org/install.html

me contacter si vous n'arrivez pas à installer jupyter via la commande :

- pip install jupyter

Une fois installé, ouvrez un invite de commande depuis le dossier dans lequel vous souhaitez conservez vos fichiers de cours (pour cela ouvrez l'explorateur de fichier windows et ouvrez votre dossier de travail, ensuite dans la barre indiquant le chemin du fichier tapez :

- cmd

puis appuyez sur "enter", cela va ouvrir l'invite de commande qui pointera vers votre dossier.

Une fois cela fait, tapez la commande :

- jupyter notebook

Cela ouvrira l'environnement de développement de jupyter (principalement utilisé pour la découverte du python ou pour la data science/ analyses statistiques)

Variables en Python

Pour rappel, en python, la déclaration d'une variable et l'affectation de sa première valeur se font en même temps et la syntaxe est la suivante :

```
variable_textuelle = “Exemple de texte”
variable_numerique = 118.218
variable_booleenne = (variable_numerique > 66)
```

Conditions en Python

En python, le SI s'écrit ***if*** et on n'a pas de Fin SI et la syntaxe est la suivante :

```
variable_numerique = 118.218
variable_booleenne = (variable_numerique > 66)
if variable_booleenne :
        print("Toutouyoutou")
elif variable_numerique <= 218 :
        print("Tatayoyo")
else :
        print("Mais c'est impossible ?!")
```

Boucles FOR en Python

En python, pour la boucle ***for*** la syntaxe est la suivante (un programme qui ne fait qu'afficher un compteur allant de 0 à 29) :

```
for compteur in range(30) :
        print(compteur)
print("Fin")
```

Boucles WHILE en Python

En python, pour la boucle ***while*** la syntaxe est la suivante (un programme qui tue un ennemi) :

```
pv_ennemi = 500
degat_joueur = 3
while pv_ennemi > 0 :
        pv_ennemi = pv_ennemi - degat_joueur
print(“l’ennemi est mort”)
```

Chap 7 : Implémentation

(traduire nos algo papiers en algo informatiques)

Trouver le nombre entre 0 et 2048

Écrivons un programme qui va choisir aléatoirement un nombre entre 0 et 2048 et nous demander de le trouver.

Lorsque nous entrerons un nombre, le programme nous répondra en indiquant si on a trouvé le bon nombre ou s'il est strictement plus grand ou plus petit que celui qu'il a généré.

Ne pas hésiter à consulter votre moteur de recherche.

Lorsqu'une erreur est levée, la LIRE... vraiment !

Ensuite si on ne la comprend pas ou qu'on ne sait pas comment la résoudre chercher sur internet (vive Stack Overflow).
Attention : Nombreuses des ressources vitales dans votre domaine seront en anglais...

Chap 8 : Listes / Tableau

Exemples

On veut calculer les moyennes des trois groupes de L1 en algo

On pourrait créer une variable *note* par élève ex: note1, note2, note3, etc.

Mais il y a une trentaine d'étudiants par groupe... ça fait long... TREEEEES long (surtout à corriger)... Donc pour éviter d'écrire une variable par note on va écrire une variable de type Liste ou Tableau (attention ce n'est pas exactement la même chose la différence sera expliquée après). Dans cette variable nous allons pouvoir stocker toutes les notes, la syntaxe est la suivante :

```
notes_grpA = [10,19,15,16,15,12,18,20,11] #etc., ici nous avons une list (un python les tableaux sont assez peu utilisés de base si bien qu'il faut les importer depuis un package)
notes_grpB = [20,12,13,10,18,17,17,10,10] #etc.
print(mean(notes_grpA)) # ici je triche car la fonction mean/moyenne n'est pas build-in mais doit être importée d'un package (nous verrons après ce que cela signifie)
print(mean(notes_grpB))
```

Tableaux

Les tableaux sont des structures de données qui servent à stocker de façon ordonnées plusieurs données de même type. Les Tableaux ont une taille fixe et ne peuvent donc contenir qu'un nombre fini et fixe de données.

Ex : une étagère de bibliothèque est comme un tableau. Elle sert à stocker des données de type Livres et elle ne peut en contenir qu'un nombre limité qui dépend de la taille qu'on lui a donné à sa création. On ne pourra pas mettre plus de livres dedans que ce que l'étagère permet de stocker.

NB : En mémoire les éléments du tableau seront stockés côte à côte en général. C'est à dire que pour rechercher des éléments du tableau dans la mémoire l'ordinateur est assez rapide et efficace.

Listes

Les listes permettent également de stocker des données de façon ordonnées mais elles n'ont pas de limites de taille (aussi en python spécifiquement, les listes peuvent contenir des données de types différents mais ce n'est pas recommandé à moins de vraiment maîtriser ce que l'on fait).

L'avantage de la liste donc est de pouvoir ajouter des éléments dès qu'on le souhaite. Cependant pour retrouver un élément dans la liste, l'ordinateur mettra plus de temps que pour un tableau. Le choix entre les deux n'a que peu d'importance sur un petit nombres de données mais si on travaille sur des grands datasets alors il vaut mieux savoir si on aura plutôt besoin d'ajouter et retirer des éléments et donc d'une liste ou si on a juste besoin d'accéder aux éléments de l'ensemble et donc du tableau.

Opérations sur les listes en python

On peut facilement afficher tous les éléments d'une liste comme suit :

```
for element in ma_liste:
    print(element)
```

Ou en utilisant un compteur et une boucle While :

```
compteur = 0
while compteur < len(ma_liste): # la fonction len() sert à récupérer la taille d'une liste
    print(ma_liste[compteur])
    compteur = compteur + 1
```

Calcul de moyenne dans une liste

Pour calculer la moyenne d'une liste il faut faire la somme de tous ses éléments pour ensuite diviser cette somme par le nombre d'éléments (la longueur de la liste):

```
def mean(ma_liste) :
        somme = 0
        for element in ma_liste :
                somme = somme + element
        resultat = somme / len(ma_liste)
        return resultat
# Voici une solution possible pour calculer la moyenne, on aurait pu aller plus vite dans le calcule de la somme des éléments de la liste car en python, une fonction sum() existe et retourne la somme des éléments d'une liste
```

Chap 9 : Turtle

Ressources utiles

https://realpython.com/beginners-guide-python-turtle/

Chap 10 : Renforcer les Acquis