Camp de jour Folie Technique
Été 2019

Informatique

Thématique 9-17 ans

POLYTECHNIQUE MONTRÉAL
Par Charly Jeffrey

Table des matières

Introduction

La semaine *Informatique* introduira les jeunes aux bases de la programmation et des concepts fondamentales de l'informatique. Ils apprendront comment manipuler les variables ainsi que les fonctions sous le langage *Python* pour créer des programmes interactifs et amusants. Ils verront également les concepts d'intelligence artificielle, cryptographie et autres au travers de diverses présentations.

La semaine est construite en supposant que les jeunes ne savent pas comment programmer; ceci n'empêchera pas les jeunes qui sont doués de se démarquer et d'en apprendre davantage.

Horaire de la semaine

<table>
<tr><th></th><th>Lundi</th><th>Mardi</th><th>Mercredi</th><th>Jeudi</th><th>Vendredi</th></tr>
<tr><td>9:00</td><td rowspan="2">Spectacle du lundi</td><td rowspan="2">Présentation: Binaire</td><td rowspan="4">Finalisation Battleship</td><td rowspan="4">Présentation: Cryp-tographie</td><td rowspan="4">Finalisation des projets</td></tr>
<tr><td>9:30</td></tr>
<tr><td>10:00</td><td rowspan="2">Introduction semaine</td><td rowspan="2">Décodeur binaire</td></tr>
<tr><td>10:30</td></tr>
<tr><td>11:00</td><td rowspan="2">Diner</td><td rowspan="2">Diner</td><td rowspan="2">Diner</td><td rowspan="2">Diner</td><td rowspan="2">Diner</td></tr>
<tr><td>11:30</td></tr>
<tr><td>12:00</td><td rowspan="2">Activité à l'extérieur</td><td rowspan="2">Activité à l'extérieur</td><td rowspan="2">Activité à l'extérieur</td><td rowspan="2">Activité à l'extérieur</td><td rowspan="2">Activité à l'extérieur</td></tr>
<tr><td>12:30</td></tr>
<tr><td>13:00</td><td rowspan="6">Introduction à Python</td><td rowspan="2">Décodeur binaire</td><td rowspan="6">Intelligence Artificielle</td><td rowspan="6">Sudoku</td><td rowspan="5">kermesse</td></tr>
<tr><td>13:30</td></tr>
<tr><td>14:00</td><td rowspan="4">Battleship</td></tr>
<tr><td>14:30</td></tr>
<tr><td>15:00</td></tr>
<tr><td>15:30</td></tr>
<tr><td>16:00</td></tr>
</table>

Conseils

Lien vers le github de la semaine.

Présentation de la semaine

Sommaire

La première activité de la semaine est simplement la présentation des activités que les jeunes feront durant la semaine. Le but est d'introduire brièvement les notions et les activités qui seront abordés pour suscité l'intérêt des jeunes. C'est également durant cette période que les jeunes se présenteront au reste du groupe.

Durée approximative	**Coût approximatif par jeune**
1h	0.00$
Niveau de difficulté (1 à 5)	**Groupe d'âge**
1	9-17ans
Matériel	**Notions abordées**
- Aucun	- Aucune

Théorie

Aucune théorie.

Manipulation

Commencez par vous présenter et rendre le groupe le plus a l'aise possible; faites une activité pour permettre à tout le groupe de se présenter rapidement. Par la suite, à l'aide de la présentation *Introduction de la semaine*, présenter aux jeunes les notions et concepts qui seront abordés durant la semaine.

Présentation suggérée

Utilisez la présentation *Introduction de la semaine*; elle contient toutes les notions à aborder. Ne vous gênez pas à sortir des notions si vous avez des *fun facts* ou autres notions qui pourraient intéresser les jeunes.

Introduction à Python

Sommaire

Cette activité a pour but d'introduire les jeunes à la programmation et au langage Python. Ils verront donc ce qu'est une variable, une fonction, certaines structures de données, les boucles et autres notions de bases.

Durée approximative	**Coût approximatif par jeune**
3h	0.00$
Niveau de difficulté (1 à 5)	**Groupe d'âge**
1	9-17ans
Matériel	**Notions abordées**
- Aucun	- Bases

Théorie

Tel que mentionné, l'activité introduira les jeunes aux notions de bases de la programmation et comment ces notions sont définies à l'aide du langage Python.

Types:

Les types sont utilisés pour définir les opérations valides pour différents *types* d'objets que l'on peut retrouver dans un programme. Voici certains types du langage Python contient:

int	Représente les nombres entiers.
float	Représente les nombres flottants.
str	Représente les chaines de caractères.
bool	Uniquement deux valeurs possibles: 0 ou 1.

Ainsi, une opérations entre deux types peut uniquement être valide si l'opération est définie par chacun des types. Exemple, il est possible d'additionner deux *int* ensemble, ou deux *string*, mais pas un *int* avec un *string*.

Variables:

Une variable peut être vu comme une petite boite qui contient une certaine valeur; on peut accéder et modifier ce que la boite contient à tout moment lors de l'exécution du programme. Autrement dit, les variables sont modifiable et permettre une programmation plus dynamique.

Fonctions:

Une fonction en informatique est l'équivalent d'une recette de cuisine: en lui fournissant les bons arguments, la fonction exécutera les lignes de codes se trouvant dans sa définition et retournera un résultat finale. Une fonction ne possède pas forcement des arguments et peut également rien retourner; cela dépend du rôle qu'on donne à la fonction.

Exemples de fonctions:

input()	Permet d'obtenir un *string* via la console
print()	Permet d'afficher un *string* à la console.
int()	Converti, si possible, l'argument en *integer*.
str()	Converti, si possible, l'argument en *string*.
len()	Obtient la taille d'une liste.

Boucles:

Les boucles sont utilisées pour répéter consécutivement une procédure plusieurs fois. Python offre deux types de boucles: les boucles *for* et *while*. La première est généralement utilisée lorsque le nombre de répétitions est connu d'avance; on utiliserait donc une boucle *for* si l'on veut parcourir chaque élément d'une liste. Les boucles *while* sont généralement utilisées lorsque le nombre de répétitions n'est pas connu; la condition d'arrêt dépend donc de la valeur d'une variable ou d'une suite de conditions dynamiques.

Listes:

Les listes en Python ne sont qu'en fait des *arrays*; il s'agit d'une structure de données dont les éléments sont enregistrés de façon ordonnée sur l'espace mémoire. Il ne faut pas confondre avec la structure *LinkedList* que l'on retrouve dans d'autres langages. En Python, une liste est un objet *mutable* et ses éléments peuvent être de type différent. Voici certaines opérations valides:

liste[i]	Obtient l'élément se trouvant à la position *i*.
liste.append(ele)	Ajoute l'élément *ele* à la fin de la liste.
liste.extend(ele1, ele2, ...)	Ajoute les éléments à la fin de la liste.
liste.insert(i, ele)	Ajoute l'élément *ele* à la position *i*.
liste.remove(ele)	Retire l'élément *ele* de la liste, si possible.

Manipulation

Le déroulement de cette activité dépend explicitement du groupe de jeunes que vous aurez; certains groupes avanceront plus rapidement que d'autres et il faudra vous adapter. Cependant, le code *HelloWorld.py* sera votre guide pour cette activité.

Commencez par ouvrir VisualCode (ou SublimeText3) et expliquer aux jeunes en quoi un éditeur de texte nous est utile lorsqu'on programme; les éditeurs de texte nous permettent d'écrire nos codes et certains ajouteront de la couleur pour faciliter la lisibilité de nos codes.

Ensuite, montrer au jeunes comment afficher du texte à la console; commencez par afficher votre nom (votre nom de camp et non votre vrai). Demandez-leur ensuite ce qu'il se passerait si vous leur donneriez votre code et qu'ils l'exécuteraient eux-même (votre nom devrait s'afficher et non le leur).

C'est généralement un bon moment d'introduire les variables. Créez une variable *nom* pour contenir le nom de la personne, puis afficher la valeur contenue dans la variable *nom*. Encore une fois, même si les jeunes exécutent votre code, le fait d'utiliser une variable ne changera pas le fait que c'est votre nom qui serait affiché s'ils exécutaient votre code. Introduisez par la suite la fonction *input* qui permet d'obtenir de l'information via la console. Ainsi, la valeur de *nom* sera spécifiée par l'utilisateur qui exécute le programme.

Pour le reste de l'activité, suivez les grandes lignes du code *HelloWorld.py*:

* Obtenir nom, age et date de naissance
* Définissez une fonction pour obtenir l'âge
* Déterminer si l'usager est majeur ou non
* Afficher un message approprié en fonction de l'âge
* etc

Vous pouvez donc improviser facilement et vous adapter à votre groupe lors de cette activité; n'hésiter pas à montrer ce que vous savez aux jeunes et assurez-vous de leur montrer comment bien coder! (commentaires, espacement, noms appropriés pour les variables, fonctions, etc)

Présentation suggérée

Aucune présentation nécessaires.

Décodeur-encodeur binaire

Sommaire

Cette activité a pour but d'introduire les jeunes au langage binaire et comment l'information est sauvegardée numériquement. Ils verront qu'est-ce qu'est la base binaire, la table *ASCII* et comment prendre un nombre, ou un caractère, pour l'encoder en binaire ainsi que comment prendre un nombre binaire et obtenir le nombre, ou caractère, original. Il est important que tous le programme de chaque jeune fonctionne puisqu'ils le réutiliseront pour la seconde partie de l'activité, soit le *Battleship*.

Durée approximative	**Coût approximatif par jeune**
3h	0.00$

Niveau de difficulté (1 à 5)	**Groupe d'âge**
3	9-17 ans

Matériel	**Notions abordées**
- Aucun	- Encodage binaire - Table *ASCII*

Théorie

Cette activité introduira relativement beaucoup de nouvelles notions aux jeunes, dont certaines qui sont des notions mathématiques. Les jeunes ne sont pas là pour avoir l'impression d'être à l'école, donc il ne faut pas trop les surcharger avec les notions mathématiques, particulièrement pour les plus jeunes.

Base:

Une base correspond au nombre de caractères qui seront utilisés pour représenter un nombre; la base décimale utilise 10 caractères qui sont les chiffres de 0 à 9. Tout nombre entier positif peut être utilisé comme base pour représenter un nombre, mais uniquement les bases 2 et 10 seront abordées par cette activité.

Soit une chaîne *string* à n caractères représentant un nombre dans la base b. Alors, sa valeur décimale d est obtenue à l'aide de l'équation suivante:

$$d = \sum_{i=1}^{n} string[i] \cdot b^{i-1} \equiv \sum_{i=0}^{n-1} string[i] \cdot b^{i}$$

Note: Les deux sommes sont équivalentes, mais l'indexation est différente; la première utilise *string[1]* pour obtenir le premier caractère tandis que la seconde utilise *string[0]*. La seconde est plus utile puisque c'est cette indexation que *Python* utilise.

Représentation binaire:

La représentation binaire permet d'exprimer tout nombre entier positif en utilisant la base 2; un nombre binaire est donc une chaîne de n caractères formée de «0» et de «1». Le nombre de caractères n nécessaire pour représenter un nombre décimale d en binaire est donné par

$$n = \lceil \log_2(d) \rceil.$$

Table ASCII:

La table *ASCII* est une catégorisation de tous les caractères existant en leur attribuant à chacun une valeur décimale unique; ainsi, en sachant la valeur numérique d'un caractère, il est possible de représenter tout caractère en binaire. Voici quelques exemples:

Caractère	Valeur numérique	Représentation binaire
a	97	01100001
b	98	01100010
A	65	01000001
B	66	01000010
?	63	00111111
2	50	00110010
3	51	00110011

Manipulation

Commencer l'activité avec la présentation *Représentation binaire* qui introduira aux jeunes les concepts nécessaires pour la réalisation de l'activité. Tel que mentionné plus haut, il ne faut pas surcharger les jeunes avec les notions mathématiques; assurez-vous toutefois qu'ils comprennent assez bien pour être capable de faire l'activité. Expliquer ensuite ce qu'ils devront réaliser: une fonction qui accepte un nombre binaire et qui retourne sa valeur décimale.

Par la suite, demander aux jeunes d'ouvrir *VisualCode* et de créer un nouveau fichier qu'ils nommeront *DecodeurBinaire.py*. Selon le groupe d'âge, les jeunes devront réaliser les fonctions suivantes:

9-10 ans : *BinairyToDecimal(); DecimalToBinairy()**
11-13 ans : *BinairyToDecimal(); DecimalToBinairy()*; BaseOneToBaseTwo()**
14-17 ans : *BinairyToDecimal(); DecimalToBinairy(); BaseOneToBaseTwo()**

Le «*» indique que les fonctions marquées ne seront pas fait par la majorité du groupe, mais uniquement par les jeunes qui ont plus de facilité et qu'ils ont fini les autres fonctions rapidement; ces fonctions doivent plus être vu comme des défis supplémentaires que ces jeunes peuvent essayer.

Une fois que les jeunes auront créé leur fichier *DecodeurBinaire.py*, la première chose à faire avec eux est d'énoncer et d'écrire les étapes principales sous forme de *pseudocode*. Une fois que le *pseudocode* est fait, laisser les jeunes essayer de programmer par eux-même le code; les 9-10 auront besoin de plus d'aide que les 14-17, mias vous pouvez toutefois leur suggérer de commencer avec le code écrit plus bas. Une fois que les jeunes auront fini leur code, ou que vous leur avez montrer la solution, assurez-vous que chacun des jeunes a un code qui fonctionne puisque leur fonction sera utilisée dans l'activité *Battleship*.

Pour les jeunes qui sont rapides, donnez-leur les autres fonctions à programmer; n'hésiter pas à leur donner des indices puisque ces fonctions sont plus complexes et demande une compréhension plus accrue de ce qu'est un nombre binaire et comment les nombres sont représentés dans une base quelconque.

Début du code

```
# Fonction pour obtenir la valeur décimal d'un nombre binaire
def BinairyToDecimal(binaire):
    # Variable qui contiendra la valeur décimale
    decimal = 0

    # Coder ici...

    # Retourne la valeur obtenue
    return decimal

# Tests de la fonction
print(BinairyToDecimal("00000010"))
print(BinairyToDecimal("00101011"))
print(BinairyToDecimal("11010101"))
```

Présentation suggérée

Utiliser la présentation *Représentation Binaire* pour cette activité.

Sources d'erreur

La première source d'erreur peut venir du fait que les jeunes seront exposés pour la première fois à une représentation des nombres dans une nouvelle base et cela peut les mélanger facilement. Assurez-vous donc qu'il comprennent bien comment les nombres sont représentés dans la base décimal, puis faites le parallèle avec la base 2 qui représente les nombres en binaire.

Une seconde source d'erreur pourrait être le fait que certains jeunes n'auraient pas vu ce qu'est une puissance en mathématique. Assurez-vous donc qu'il comprennent bien ce qu'une puissance représente. Vous pouvez faire le parallèle entre la multiplication qui simplifie l'écriture de plusieurs additions consécutives et la puissance qui simplifie l'écriture de plusieurs multiplication consécutives. Exemple:

$$\underbrace{3+3+3+3}_{\text{Quatre '3' additionnés}} = 4 \times 3$$

$$\underbrace{3 \times 3 \times 3 \times 3}_{\text{Quatre '3' multipliés}} = 3^4$$

Battleship

Sommaire

Cette activité est la suite de celle du décodeur binaire que les jeunes auront fait juste avant. Elle utilise ce que les jeunes ont vu et fait pour mettre en pratique leur compétences. La mise en situation de l'activité est la suivante: la position de bateaux ennemies a été reçu, mais l'information est encodé en binaire; il faut donc la décoder pour être capable de l'utiliser.

Durée approximative	**Coût approximatif par jeune**
2h	0.00$
Niveau de difficulté (1 à 5)	**Groupe d'âge**
3	9-17ans
Matériel	**Notions abordées**
- Aucun	- Fichier

Théorie

Durant cette activité, les jeunes apprendront à lire un fichier *texte* à l'aide de *Python* et d'en retirer l'information qu'il contient. Il faut donc que les jeunes comprennent bien comment un fichier *texte* est sauvegardé numériquement.

Fichier texte:

Les fichiers de type *texte* sont très utilisés pour enregistrer de l'information tel que des notes, des directives, etc. Ces fichiers ont deux propriétés importantes: le nombre de lignes qu'ils contiennent ainsi que le nombre de caractères maximal par ligne.

Python et les fichiers:

Il est très facile de lire des fichiers *.txt* à l'aide de *Python* et cela peut se faire en quelques étapes:

1 - Ouvrir le fichier à l'aide de la fonction $f =$ `open("path_to_file\FILE_NAME", "r")`
2 - Obtenir toutes les lignes du fichier avec $lines = f.$`readlines()`
3 - Boucler sur chaque lignes avec `for` *line* `in` *lines:*
4 - Utiliser la variable *line* pour manipuler les données de la ligne

Ainsi, la boucle passera à la variable *line* chacune des lignes du fichier, une à une.

Manipulation de *string*:

Python permet de facilement manipuler des objets de type *string* pour les modifier comme bon nous semble. Les méthodes de la classe *string* seront très utiles pour l'activité:

`Obj.split(c)`	Permet de séparer la chaîne «*Obj*» à chaque fois que le caractère «*c*» est rencontré. Espace par défaut.
`Obj[i:j:p]`	Permet d'obtenir la sous-chaîne de «*Obj*» débutant au i^e caractère et se terminant au j^e, en faisant des saut de p. p = 1 par défaut.

Par exemple, soit la chaîne «*var = "Allo toi"*», alors

```
   var.split()   = ["Allo", "toi"]
var.split('o')   = ["All", "t", "i"]
     var[0:4]    = "Allo"
    var[0:7:2]   = "Al o"
     var[::-1]   = "iot ollA"
```

Il sera important de faire la distinction entre ce qui est retourné par chacune des méthodes: la méthode *.split()* retourne un tableau des éléments qui se retrouvent entre le caractère de séparation et l'indexage *Obj[i:j:p]* retourne un *string* qui contient les éléments respectant les indices fournis.

Matériel

Aucun matériel n'est nécessaire pour cette activité, mais vous aurez besoin des répertoires/fichiers suivant:

/Battleships	Répertoire qui contient les fichiers nécessaire à l'activité.
/Battleships/Battleship.py	Programme *python* pour générer de nouveaux fichiers encryptés.
/Battleships/Battleships/Equipes	Répertoire qui contient les fichiers encryptés en binaire.
/Battleships/index.html	Interface de jeu pour vérifier la validité des décryptions de chaque équipe.

Manipulation

Il y a plusieurs manipulations à faire avant le début de l'activité:

Création des fichiers encryptés:

Pour générer les fichiers que les jeunes devront décrypter, il faut utiliser le programme *Battleship.py* via la console de la manière suivante:

$$\$\ python3\ Battleship.py\ < nombre\_de\_bateaux > < nombre\_d\_equipe >$$

Cette commande va créer les répertoires et fichiers nécessaire à l'activité; si ces répertoires existent déjà, les fichiers existants seront alors écrasés. L'argument *nombre_d_equipe* dictera combien de fichiers encryptés aléatoires seront créés tandis que *nombre_de_bateaux* décidera du nombre de bateaux que chaque fichier contiendra. Normalement, une seule utilisation du programme *Battleship.py* serait nécessaire pour tout l'été, mais vous pouvez toujours générer de nouveaux fichiers encryptés comme bon vous semble. Vous pouvez également lancer le programme sans l'utilisation du terminal, ce qui va lancer l'application avec 30 équipes et 15 bateaux.

Distributions des fichiers:

Cette partie est cruciale puisque sans celle-ci, les jeunes ne peuvent pas réaliser l'activité. Il s'agit de leur distribuer les fichiers encryptés sur leur session. Ceci peut rapidement être fait à l'aide du logiciel *NetSupport* qui vous permet d'échanger des documents entre votre ordinateur et ceux de la classe. Notez que si vous générer de nouveaux fichiers avec *Battleship.py*, vous devez les distribuer à nouveaux sinon il y aura une incohérence entre les fichiers que les jeunes décryptent et ceux que l'interface visuelle a.

Comment utiliser l'interface:

Pour lancer l'interface visuelle, ouvrer simplement le fichier *index.html* dans le répertoire */Battleships*; une fenêtre *chrome/firefox* devrait s'ouvrir. L'interface est séparer en deux: la grille de jeu et la partie pour sélectionner une équipe et tester/montrer la solution. Pour tester les coordonnées qu'un jeune vous donne, sélectionner son équipe dans la bar de sélection et cliquer sur le bouton *ok*. Par la suite, dans la grille de jeu, cliquer sur les cases correspondantes aux coordonnées qu'il vous donne et appuyer sur le bouton *Vérifier solution*; les coordonnées correctement décryptées auront un *checkmark* vert tandis que les mauvaises réponses auront un *X* rouge. Si vous voulez montrer la solution, vous n'avez qu'à sélectionner une équipe, appuyer sur *ok* puis appuyer sur *Montrer solution*. Si vous lancer le programme *Battleship.py* pendant que l'interface fonctionne, vous devez fermer la fenêtre puis relancer l'application *index.html* mettre à jour les coordonnées.

Les prochaines manipulations se font durant l'activité, soit avec les jeunes.

Explication de l'activité:

Assez simple, expliquez aux jeunes ce qu'ils réaliseront durant cette activité.

Explication des méthodes *.split()* et *indexage*:

Cette étape est assez importante puisqu'elle introduit aux jeunes les notions nécessaires pour qu'ils soient en mesure de lire et manipuler les fichiers *.txt* encryptés. Faites un exemple assez simple avec eux:

1 - Faites leur créer un fichier *.txt* et qu'ils écrivent n'importe quoi à l'intérieur.
2 - Montrez leur comment lire ce fichier à l'aide de la méthode définie plus haut.
3 - Montrez leur comment obtenir une sous-chaîne d'un *string* quelconque.
4 - Montrez leur comment séparer un *string* à l'aide de la méthode *.split()*.

Commencer l'activité:

Une fois que les fichiers ont été distribués et que l'exemple sur la lecture d'un fichier ainsi que les méthode pour modifier un *string* sont bien compris par les jeunes, vous pouvez commencer l'activité. La première étape est d'ouvrir l'interface visuelle -*index.html*- et expliquer aux jeunes ce qu'ils auront à faire. Vous pouvez ouvrir un des fichiers encryptés, peu importe lequel, pour s'assurer que les jeunes comprennent bien ce qu'ils contiennent.

Lorsque les jeunes commencent à programmer, votre rôle sera simplement de les superviser et de répondre aux questions; si une question revient souvent (2 à 3 fois), faites une pause pour l'expliquer au groupe. Lorsqu'un jeune pense avoir réussi, selectionner son numéro d'équipe dans l'interface et tester les coordonnées qu'il a obtenu; utilisez l'option *Vérifier solution* et non pas *Montrer solution*.

Lorsqu'un grand nombre de jeunes ont fini, ou pense avoir fini, et que la période est presque terminée, vous pouvez selectionner chacune des équipes dans l'interface et utiliser l'option *Montrer solution*; les jeunes pourront donc vérifier si leur solution est bonne et noter la bonne dans le cas contraire pour tester leur code par eux-même par la suite.

Sources d'erreur

Une erreur possible serait que les fichiers solution que l'interface utilise ne soient pas les même que les fichiers que les jeunes utilisent; dans ce cas, vous devez redistribuer les fichiers dans le répertoire */Battleships/Battleships/Equipes* aux jeunes.

Intelligence Artificielle

Sommaire

Activité qui introduira les jeunes au concept d'intelligence artificielle et comment un ordinateur peut «apprendre» à faire certaines tâches spécifiques à l'aide du *Machine Learning*. Les jeunes ne feront pas de programmation, mais testeront quelques jeux/applications utilisant des AI/ML.

Durée approximative	**Coût approximatif par jeune**
3h	0.00$
Niveau de difficulté (1 à 5)	**Groupe d'âge**
0	9-17ans
Matériel	**Notions abordées**
- Aucun	- *Machine Learning* - *Intelligence Artificielle*

Théorie

Intelligence Artificielle:

Une intelligence artificielle est simplement un programme qui exécute une certaine tâche automatiquement en suivant une procédure précise. Ainsi, le programme est donc capable de simuler un niveau d'intelligence pour accomplir une tâche précise. Une intelligence artificielle peut donc être responsable de trier les courriers que vous recevez, de déterminer si une image contient un chat, de déplacer des personnages dans un jeu, etc.

Machine Learning:

Le *Machine Learning*, ou apprentissage machine, est une application de l'intelligence artificielle qui permet à une machine d'apprendre comment réaliser une tâche par elle-même, sans explicitement lui fournir de code. Ainsi, en lui fournissant les données nécessaires, une machine peut utiliser ces données pour apprendre et améliorer graduellement sa compétence à réaliser une tâche. Les applications du *Machine Learning* sont très nombreuses et il s'agit d'une branche de l'informatique en constante évolution.

Réseau de neurones:

L'idée dernière le *Machine Learning* est d'utiliser un réseau de neurones qui simule un cerveau humain; chaque neurone du réseau est connectée à d'autre et ces connections peuvent être de différentes intensités. Ainsi, lorsqu'un AI apprend à faire une tâche en utilisant du *Machine Learning*, il suffit de modifier les connections entre les neurones du réseau pour tenter d'augmenter l'efficacité de ce réseau à accomplir la tâche demandée. Un réseau est composé de trois couches: la couche d'entrée, la couche cachée et la couche de sortie.

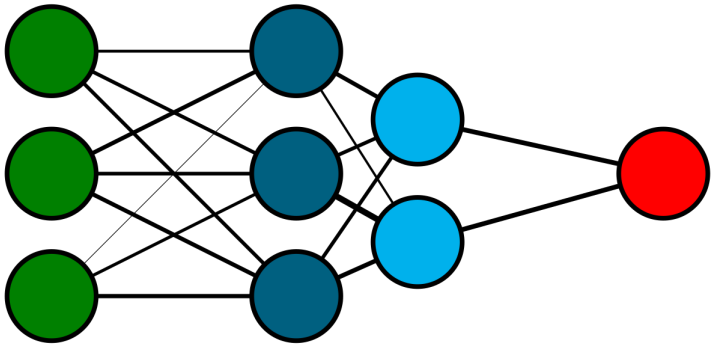
Les couches d'un réseau:

La couche d'entrée d'un réseau est simplement l'information qu'on envoie au réseau. Il pourrait donc s'agir de l'intensité de chaque pixel d'une image, par exemple.

La couche cachée est la plus importante d'un réseau puisque c'est celle-ci qui déterminera la réponse que le réseau donnera selon l'information qu'il a reçue en entré. Cette couche est normalement composée de plusieurs autres couches de neurones et le nombre de neurones par couche varient selon le réseau; il n'y a pas vraiment de règles claires définissant les dimensions de cette couche.

La dernière couche du réseau est la couche de sortie; il s'agit de la réponse du réseau suite à l'information qu'il a reçue. Exemple, on envoie à un réseau une image et on veut savoir si cette image contient un chat, alors la réponse du réseau sera soit oui ou non. La taille de cette couche dépend explicitement de la tâche que le réseau doit accomplir; l'exemple du chat nécessite uniquement un neurone.

Exemple de réseau:



Soit le réseau représenté par la figure ci-dessus. Sa couche d'entrée est composée des trois neurones verts, sa couche cachée est composée d'une première couche de trois neurones suivit d'une seconde couche de deux neurones puis le réseau se termine par un unique neurone rouge. Les lignes reliant les neurones représentent avec quelle intensité deux neurones sont connectés; plus la ligne les reliant est épaisse, plus intense est la connection.

Manipulation

Commencer par suivre la présentation *Intelligence artificielle*. Lorsque cette présentation est terminée, vous pouvez laisser les jeunes jouer au jeu *iKart* qui testera si les jeunes sont capable de battre un AI à un jeu de course. Cette AI a appris par elle-même à jouer à l'aide d'un réseau de neurones.

Présentation suggérée

La présentation *Intelligence artificielle* est nécessaire à cette activité.

Sudoku

Sommaire

Cette activité permettra aux jeunes de créer leur propre jeu Sudoku via la console en utilisant ce qu'ils ont vu durant la semaine. Il s'agit donc d'une activité qui n'introduit pas de nouveau concept, mais met en pratique ce que les jeunes ont appris au cours de la semaine.

Durée approximative	**Coût approximatif par jeune**
3h	0.00$
Niveau de difficulté (1 à 5)	**Groupe d'âge**
3	9-17ans
Matériel	**Notions abordées**
- Aucun	- Récapitulatif

Théorie

Sudoku est un jeu assez simple où il faut remplir une grille 9x9 en utilisant les chiffres de 1 à 9 en respectant les trois contraintes suivantes:

1) Un chiffre ne peut pas se retrouver deux fois dans la même rangé.
2) Un chiffre ne peut pas se retrouver deux fois dans la même colonne.
3) Un chiffre ne peut pas se retrouver deux fois dans la même sous-grille 3x3.

Les jeunes programmeront une version simplifiée de ce jeu à l'aide de Python et l'interaction se fera avec la console.

Manipulation

Commencez d'abord par expliquer comment jouer au Sudoku et assurez-vous bien que les jeunes comprennent les règles du jeu. Vous pouvez ensuite commencer la programmation. Le code complet se trouve dans le répertoire *Sudoku*.

0) Créer une grille de jeu

L'étape 0 est d'expliquer au jeune la structure qui sera utilisée pour représenter une grille de jeu; il s'agit simplement d'un *array* qui contient 9 *arrays*, chacun représentant une rangée de la grille de jeu. Exemple:

$$
\begin{matrix}
[[0,0,0,8,0,5,0,1,3]\ , \\
[0,0,0,2,0,3,6,0,0], \\
[6,0,0,0,9,0,2,0,4], \\
[0,0,0,0,0,0,0,0,5], \\
[0,4,0,1,0,0,7,0,6], \\
[2,5,6,3,0,4,8,9,0],
\end{matrix}
$$

$$[5, 9, 0, 0, 0, 7, 1, 0, 2],$$
$$[1, 0, 2, 0, 8, 0, 4, 7, 0],$$
$$[0, 0, 4, 9, 1, 0, 0, 3, 8]]$$

représente une grille de jeu dont les 0 sont les cases initialement vides. En fait, la grille de jeu possède une troisième dimension qui peut uniquement avoir les valeurs *true/false* pour déterminer si la case était initialement vide; cette troisième dimension n'est toutefois pas affichée lors d'une partie.

1) Afficher la grille de jeu

La première étape du code est de créer une fonction pour afficher clairement la grille de jeu; par exemple, le caractères '|' peut être utilisé comme séparateur pour les colonnes, '_' comme séparateur pour les rangés et ainsi pouvoir afficher des cases. Évitez d'utiliser des boucles pour l'affichage avec les 9-10 ans puisque cela risque de les mélanger; utilisez votre jugement pour les 11-13 ans.

2) Placer un chiffre dans une case

La seconde étape est de créer une fonction pour obtenir la case dans laquelle le joueur veut placer un certain chiffre. Cette fonction doit donc vérifier si la case est libre et si le chiffre que le joueur veut placer peut se retrouver dans cette case; la fonction vérifiera donc si ce chiffre ne se trouve pas déjà dans la même rangé, colonne ou sous-grille. Le tout se fait dans un boucle dont la condition est initialement toujours vraie.

Commencez d'abord par obtenir la case dans laquelle le joueur veut jouer et placer le chiffre désiré dans cette case, sans vérifier si c'est un mouvement valide. Ajoutez ensuite la partie qui fait la vérification et placer le chiffre dans la case uniquement si cela est valide. Notez que uniquement les cases qui étaient initialement vides peuvent être modifiées lors d'une partie par le joueur, donc le joueur peut modifier une case qu'il a déjà remplie, mais pas une case qui contenait initialement un chiffre.

3) Condition d'arrêt

Une fois que la fonction pour placer un chiffre dans la grille est terminée, il faut maintenant déterminer la condition d'arrêt de la boucle de jeu. Une manière de faire cela est de déterminer le nombre de cases initialement vides que le joueur doit remplir. à chaque fois que le joueur place un chiffre dans une case qui est vide, il suffit alors de diminuer de 1 le nombre de cases vides. Ainsi, lorsque le nombre de cases vides atteint 0, la partie est terminée et le joueur a gagné.

4) Résoudre un sudoku

Cette étape est optionnelle et dépend du temps qu'il vous reste pour cette activité: il s'agit d'écrire une fonction pour obtenir les solutions possibles pour une certaine grille de jeu initialement incomplète. La fonction utilisée est *récursive*, ce qui signifie qu'elle fait appelle à elle-même lors de son exécution. La définition de cette fonction se trouve également dans le répertoire *Sudoku*.

Présentation suggérée

Aucune présentation nécessaire.

Sources d'erreur

Le programme est plutôt court et cette activité n'introduit pas de nouvelles notions; les erreurs qui peuvent survenir sont donc des erreurs que les jeunes auront déjà rencontrées lors des autres activités.