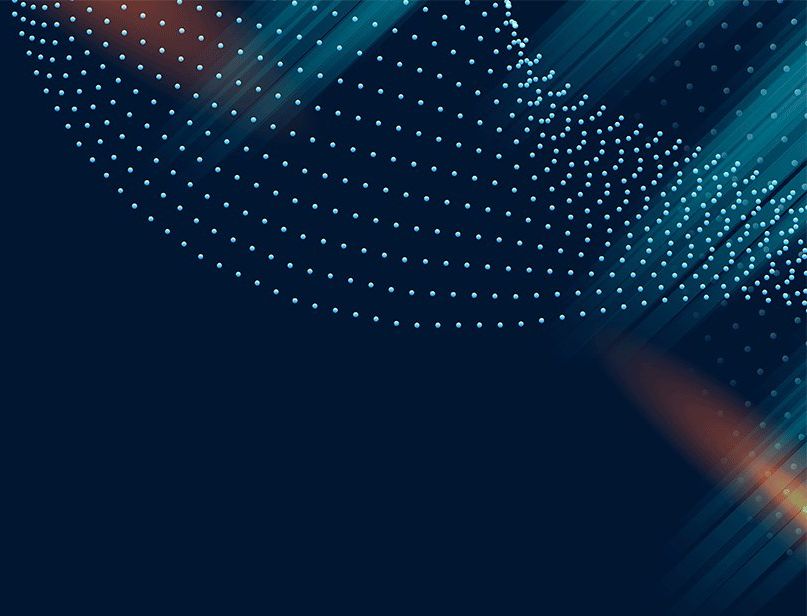


Python (+ git)

Introduction

Les bases de Python

- variables
- conditions (if)
- boucles (while, for)
- fonctions
- listes / dictionnaires



Les bases de Git

- `git init`
- `git add fichier.txt`
- `git add .`
- `git commit -m "Un exemple de commit"`
- `git status`
- `git push`

Qu'est ce qu'une
branche ?

Que signifie « merge » /
« merge request »

Exercice 1 : Création de ton personnage

- **Objectifs** : variables, print, input.

Écrire un programme qui demande :

- le nom du personnage
- sa classe (Guerrier, Mage, Voleur, etc.)
- son niveau de départ -> Le niveau est défini à 1 dans le programme
- ses points de vie de base et sa force
- Puis affiche :

- Ton personnage <Nom> est un <Classe> de niveau <Niveau> avec <PV> PV et un niveau de force de <Force>
- Push le code sur Git lorsque cette « feature » est fonctionnelle

Exercice 2 : Combat simple contre un squelette

- **Objectifs** : conditions, comparaisons, librairies

Écrire un programme où :

- Le squelette a 10 PV (Points de Vie)
- Le joueur frappe une seule fois
- Le coup généré est calculé entre 5 et 15 (`random.randint`)
- Si le joueur inflige ≥ 10 dégâts :
 - Afficher "Squelette vaincu !"
 - Faites monter votre personnage de 1 niveau et afficher le niveau
- Sinon → Afficher "Le squelette se moque de toi..."
 - Push le code sur Git lorsque cette « feature » est fonctionnelle

Exercice 3 : Boucle de combat contre un Golem

- **Objectifs** : Boucles, décrémentation, logique de répétition

Écrire un programme où :

- Le Golem a 50 PV (Points de Vie)
 - Le personnage est monté de niveau !
Il possède 3 nouveaux sorts :
« Coup d'épée » inflige 10 à 14 dégâts,
« Lancer de dague » inflige 2 à 22 dégâts
« Boule de feu » inflige soit 0 (raté), soit 20 dégâts (`random.choice`)

À chaque tour :

- Le programme demande quel sort lancer (taper 1, 2 ou 3
-> Faire correspondre avec vos sorts)
- Afficher le sort lancé
- Réduit les PV du Golem
- Le Golem vous frappe de 7 dégâts
- Affiche les PV restants de chacun

La boucle continue jusqu'à :

- Si Points de Vie du Golem inférieur ou égal à 0 :
 - Afficher "Le Golem est KO !"
- Sinon → Le combat continue !

Push le code sur Git lorsque cette « feature » est fonctionnelle

Exercice 4 : Qu'est ce que j'ai dans mon inventaire ?

- Objectifs : Listes
- Créer une liste :
inventaire = ["Potion de soin", "Potion de mana", "Pièce d'or"]
- Actions :
 - Afficher la taille de l'inventaire
 - Ajouter l'Arme Ultime gagné dans le coffre
 - Supprimer (buvez) la potion de soin
 - Afficher l'inventaire final
- Push le code sur Git lorsque cette « feature » est fonctionnelle

Exercice 5 : Le coffre piégé

- **Objectifs** : boucle + conditions
- Pour ouvrir un coffre :
 - Le code magique est un nombre entre 1 et 20 (choisi avec `random.randint`)
 - Le joueur doit deviner, mais il a 3 essais maximum, sinon... Le coffre mange son Or !

Chaque tour :

- Demander un nombre
- Afficher "Trop haut", "Trop bas" ou "Bravo ! Vous avez trouvé l'Arme Ultime !"
 - Ajouter « l'Arme Ultime » à votre inventaire
- Si le nombre d'essai est supérieur à 3, afficher « Le coffre a mangé votre Or ! »
 - Supprimer « Pièce d'Or » de votre inventaire

Push le code sur Git lorsque cette « feature » est fonctionnelle

Exercice 6 : Calcul de dégâts

- **Objectifs** : écrire et appeler une fonction
- Créer une fonction « calcul_degats », qui prend en paramètres « Force » et « Arme »
- Il faut donc demander la puissance de son arme pour calculer ses dégâts
- Actions :
 - Créer la fonction
 - Lui donne la formule de calcul de dégâts
 - L'appeler avec les bons parametres
 - Afficher les dégâts de votre personnage :
 <Nom> peut infliger <calcul_degats> de dégâts
 - Push le code sur Git lorsque cette « feature » est fonctionnelle

Exercice 7 : Exploration du donjon

- **Objectifs** : combiner votre code pour tout ce qui précède
- Le joueur doit franchir 3 salles :
 - Combattre un Squelette (if)
 - Trouver le code d'une porte (while + random)
 - Golem → boucle de combat (while)

Afficher à la fin : **Félicitations, tu as terminé le donjon !**

Push le code sur Git lorsque cette « feature » est fonctionnelle