🔥 TFA 4TTR: Générateur de personnages D&D - 2025

Dans ce projet, vous allez utiliser Python pour créer un générateur de personnages pour le jeu de rôle Donjons et Dragons. Vous allez demander à l'utilisateur d'entrer des informations sur le personnage, générer des statistiques de personnage au hasard, calculer des modificateurs et des points de vie, et effectuer des tests de caractéristiques.

À la fin du programme, vous devez afficher toutes les caractéristiques du personnage : nom, race, classe, caractéristiques avec modificateur, points d'XP, niveau...

Consignes générales

- **Téléchargez les versions originales et personnelles** de votre projet dans le dossier `TFA 2025 2025\`.
- Nom du fichier principal: `tfa_2025_[prenom].py` (exemple: `tfa_2025_lara.py`)
- Écris les fonctions dans le fichier [prenom]_utils.py (exemple: `lara_utils.py`) et importe ce fichier dans ton programme principal.
- Inclue 3 exemples d'exécution de votre code (copie-colle le contenu complet de la console dans des fichiers textes): `exam\_2024\_[prenom]\_run\_x.txt` (où x vaut 1, 2 ou -3
- Utilise **Thonny IDE** pour réaliser ton projet. Tu peux utiliser **Visual Studio Code** s'il est configuré pour Python.
- Utilise des noms de variables corrects.
- Commentez votre code (ce qu'il fait, pourquoi vous avez écrit telle ligne de telle manière...).
- Le code doit fonctionner au démarrage du programme. C'est-à-dire qu'il ne peut pas y avoir d'erreur de syntaxe au moment où on l'exécute. Pénalité : 50% des points en moins avant même de corriger la suite de l'exercice.
- Déposez vos fichiers sur le site https://fichiers.ttrinfo.be dans le dossier `Examen 2025\Python\v1`.

Aides

- Pour quitter un programme Python, utilisez la fonction `exit()`.
- Pour vérifier si une chaîne de caractères ne contient que des chiffres, utilisez la méthode `isdigit()`. Ex. : `"1234".isdigit()` → `True` ou `ma_variable.isdigit()`.

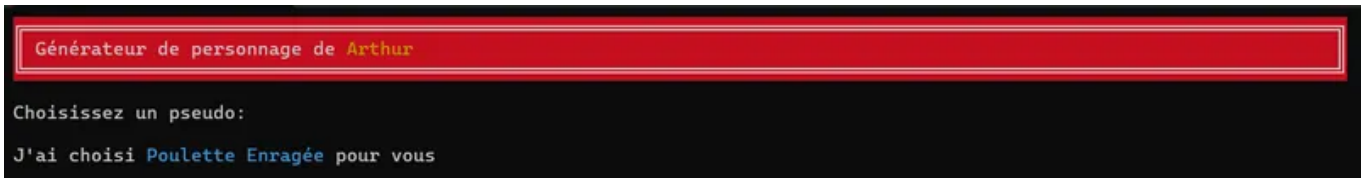
Étape 1: Demander un nom

Demandez à l'utilisateur d'entrer un pseudo pour le personnage. Si le nom est vide, générez un pseudo en sélectionnant un animal et un adjectif au hasard parmi ces deux listes :

```
animaux = ["Souris", "Lapin", "Licorne", "Dragon", "Raton", "Biquette", "Cheval"]
adjectifs = ["Intrépide", "Sauvage", "Magique", "Badass", "Boloss", "Baltringue"]
```

(Ces listes sont propres à ton projet)

Exemples :



Etape 2: Choisir une race

Affichez une liste de races possibles. Les races sont inclues dans ton fichier personnel.

L'utilisateur doit choisir parmi cette liste en tapant un numéro correspondant à l'option. Vérifiez qu'il a bien entré un nombre et que ce choix est valide. Si c'est le cas: enregistrez-le, sinon recommencez. Répétez tant qu'aucune race n'a été sélectionnée (d).

La sortie à l'écran doit ressembler à ceci:

Etape 3: Choisir la classe

Demandez à l'utilisateur d'effectuer son choix de classe. Enregistrez son choix. Les classes sont propres à ton projet. Ajoutez une ligne "Au hasard" (manuellement) au menu.

Vérifiez qu'il a bien entré un nombre et que ce choix est valide. Si c'est le cas: enregistrez-le, sinon recommencez. Répétez tant qu'aucune option valide n'a été sélectionnée.

L'affichage à l'écran doit correspondre à ceci :

💥 **BONUS**: Il est possible de créer **une** fonction qui permet d'effectuer un choix parmi une liste d'options. Cette fonction recevrait la liste de choix en paramètre et retournerait le choix de l'utilisateur:

```
def menu(liste_choix):

    # ...

    return choix
```

Tu peux créer cette fonction et l'utiliser pour grandement simplifier le code de la sélection de la race et de la classe.

Étape 4: Générer les 6 caractéristiques au hasard

Créez une fonction pour effectuer un lancer de dé

Créez une fonction qui prend en paramètre le nombre de faces du dé et renvoie un nombre aléatoire entre 1 et ce nombre (inclus). Vous allez utiliser cette fonction pour générer les caractéristiques et effectuer les tests de caractéristiques.

Générez les 6 caractéristiques

Les caractéristiques de base d'un personnage de Donjons et Dragons sont :

- **FOR** : Force
- **DEX** : Dextérité
- **CON** : Constitution
- **INT** : Intelligence
- **SAG** : Sagesse
- **CHA** : Charisme

Vous pouvez stocker ces caractéristiques **dans 6 variables**. Dans un premier temps, pour chaque caractéristique, effectuez un lancé de `d20`.

Étape 5: Boost de caractéristiques

Ton personnage a droit à un **boost de caractéristique** généré **au hasard** en fonction de sa race. Ces bonus sont appliqués pour chaque race de ton fichier perso. La 1ère race de ta liste reçoit un bonus de force, la 2e de dextérité... et ainsi de suite dans l'ordre suivant:

N°	Caractéristique
1	Force
2	Dextérité
3	Constitution
4	Intelligence
5	Sagesse

Par exemple:

```
1 | races = ["Humain", "Nain", "Elfe", "Halfelin", "Dragon"]
```

- **Humain** --> **1d4** points à sa **force**
- **Nain** --> **1d4** points à sa **dextérité**.
- **Elfe** --> **1d4** points à son **constitution**.
- **Halfelin** --> **1d4** points à sa **Intelligence**.
- **Dragon** --> **1d4** points à sa **Sagesse**.

Tu t'en sortiras très bien avec une série de `if`, `elif` et `else` par exemple.

Incrémente la caractéristique correspondante et affiche un message qui indique ce bonus. Pas besoin de sauvegarder le boost pour plus tard.

Exemples:

```
Votre Humain reçoit un bonus de 2 FOR.
```

(Donc, avant le boost: FOR = 12, après FOR = 14)

```
Votre Nain reçoit un bonus de 2 DEX.
```

(Donc, avant le boost: DEX = 13, après DEX = 15)

Étape 7: Modificateur de caractéristique

Le **modificateur de caractéristique** dans *Donjons & Dragons* est un nombre qui représente à quel point une caractéristique influence les actions du personnage.

Il est calculé de la manière suivante **à partir de la valeur brute** de la caractéristique:

- soustraire 10 de la valeur de la caractéristique
- diviser le résultat par 2
- arrondir vers le bas

Exemple :

- Une Force de 16 donne un modificateur de **+3**
- Une Intelligence de 9 donne un modificateur de **−1**

À quoi ça sert ?

Le modificateur est utilisé pour :

- Ajouter ou retirer des points aux **jets de dés** (attaque, compétence, sauvegarde…)
- Déterminer la **puissance ou l'efficacité** d'une action

Créez la fonction qui permet de calculer un modificateur sur base de la valeur d'une caractéristique. Cette fonction sera utilisée plus tard.

Étape 8: Calculer les points de vie (PV)

Normalement, les points de vie d'un personnage sont calculés sur base de sa classe et de sa Constitution. Pour simplifier, vous pouvez supposer que tous les personnages commencent avec un nombre de points de vie égal à **10** plus leur **modificateur de Constitution** plus le résultat d'un **d4**.

Affichez les points de vie:

```
Points de vie: 17
```

Étape 9: Points d'expérience (XP)

Pour calculer les points d'expérience (XP) de votre personnage, additionnez toutes ses caractéristiques et multipliez le tout par 6. Ajoutez ensuite un multiple de 100 au hasard compris entre 100 et 1000.

Affichez le résulat:

```
Points d'expérience: 350
```

Étape 10: Niveau

Le niveau de votre personnage est calculé en fonction de son nombre de points d'expérience:

Niveau	XP requis
1	0
2	300
3	900
4	2700
5	6500
6	14000

Affichez le niveau de votre personnage:

```
Niveau: 2
```

Étape 11: Affichage

Affichez votre personnage selon le modèle ci-dessous.

Le chiffre entre parenthèses est le modificateur de la caractéristique qui est calculé par la fonction créée précédemment. Si le modificateur est positif, il faut le précéder du signe "+" (astuce : utilisez des f-strings).

Étape 12: Tests de caractéristiques

Un **test de caractéristique** permet de tester le talent inné et l'entraînement d'un personnage ou d'un monstre face à un défi et détermine le résultat d'une **action**.

Pour chaque test de caractéristique, le Maître du Jeu choisit une des six caractéristiques et la difficulté de l'action à accomplir, qui est représentée par un **degré de difficulté** (DD).

Pour effectuer votre test de caractéristique, effectuez un lancé de `d20`, et ajoutez le **modificateur de la caractéristique testée**. Si ce total est égal ou supérieur au DD, le jet est un **succès**, la créature a réussi à surmonter le défi. Sinon, c'est un **échec**.

Par exemple, si votre personnage a **14 (+2)** de **FOR**, il peut réussir une tâche qui demande 16 de FOR s'il effectue un lancé de dé de 14 ou plus.

Les actions que vous devez tester sont fournies dans le fichier que vous avez téléchargé.

Ces actions sont représentées sous forme d'une liste d'objets Python en respectant les noms de propriétés suivants :

```
actions = [
    {
        "intitule": "Soulever un rocher", # nom de l'action
        "caracteristique_testee": "FOR",  # caractéristique testée
        "degre_difficulte": 14,           # degré de difficulté
        "pv": 0,                          # nombre de points de vie à perdre en cas d'échec
        "succes": False,                  # test réussi ou pas?
        "xp": 150                         # xp gagnée
    },
    #...
]
```

Pour cette étape, créez une fonction `test_caracteristique(action)` où

- action : l'action à tester

Dans cette fonction, vous pouvez mettre l'action à jour.

Modification à apporter à l'action:

- la valeur du champ `succes` en fonction du résultat du test

Cette fonction doit **afficher le résultat** du test et retourner la valeur `True` si le test a réussi ou `False` si le test a échoué.

Voici des exemples de tests à réaliser. Pour chaque test vous avez son intitulé, la caractéristique testée, la valeur minimale pour réussir (degré de difficulté) et le nombre de points de vie en jeu:

Action	Caractéristique	DD	XP	PV
"Soulever un rocher"	FOR	14	250	0
"Escalader la falaise"	DEX	13	450	10
"Encaisser un coup de poing"	CONST	1	100	3

Effectuez une boucle pour réaliser tous les tests demandés.

Exemple:

Étape 13: Comptabilisation des résultats

Vous allez maintenant afficher un résumé des résultats des différents tests effectués à l'étape précédente.

Affichez le **nombre de tests** qui ont réussi et qui ont échoué.

XP

Affichez le nombre total de **points d'expérience gagnés** (Rappel: les points d'expérience ne sont gagnés que si le test a été réussi).

Niveau

Maintenant que vous connaissez le nombre de points d'XP final, affichez le **niveau final** de votre personnage :

```
Votre personnage est maintenant au niveau 4.
```

Si votre personnage a gagné des niveaux affichez le nombre de niveaux gagnés:

```
Félicitations, votre pesonnage a gagné 1 niveau(x)"
```

ou

```
Dommage, votre personnage n'a pas gagné de niveaux"
```

PV

Comptabilisez également les **points de vie** de votre personnage. Chaque action a un nombre de points de vie qui lui est associé (`pv`). Si votre personnage a échoué au test, il perd les points de vie correspondants.

Nouvel état

Affichez maintenant le nouvel état de votre personnage en vous basant sur les données et l'exemple suivants:

État en fonction des PV mis à jour:

```
< 10 : Affaibli
< 5  : état critiqe
< 0  : inconscient
<= -10: mort
```

Étape 14: Génération des caractéristiques V2

En réalité, dans D&D, les points des caractéristiques ne sont pas générés à l'aide d'un d20 (dé à 20 faces).

Au lieu de cela, pour chaque caractéristique, il faut effectuer **un jet de 4d6** (lancer 4 dés à 6 faces) et **additionner les 3 meilleurs dés**.

Vous allez donc créer une fonction `genererCaracteristique()` afin de prendre cela en compte.

Commentez l'ancien code (ne le supprimez pas) et suivez ces conseils:

- Stocker les 4 lancés dans un tableau (une liste)
- Si vous triez la liste par ordre décroissant, il vous suffit de garder les 3 premiers lancés (autrement dit, de créer une sous liste contenant les 3 premiers éléments)
- Il existe une fonction en Python qui permet de calculer la somme des éléments d'une liste en 1 instruction.

© 2025 Mr. Lambrechts https://v2.ttrinfo.be 9 | 14

Pour chaque caractéristique, utilisez cette fonction au lieu de lancer un d20.

La fonction doit afficher les données de chaque étape comme suit:

```
Lancés:[3, 1, 1, 3]
Triés:[3, 3, 1, 1]
Meilleurs:[3, 3, 1]
Total: 7
```

Ligne 1: Liste des 4 lancés Ligne 2: Les 4 lancés triés Ligne 3: Les 3 meilleurs lancés Ligne 4: la somme des 3 meilleurs lancés

Étape 15: Attribution de pièces d'or

Chaque joueur commence son aventure avec un nombre de pièces d'or déterminé au hasard en fonction de son charisme. Pour calculer le nombre de pièces d'or initial, il faut effectuer un jet de **20d6** (=4 lancés de dés à 6 faces).

Étape 16: Inventaire initial

Au début de l'aventure, le joueur **reçoit** automatiquement **deux items de départ**, choisis parmi une liste prédéfinie selon la race du personnage. L'inventaire du joueur est limité à un **maximum de 5 items**.

La liste des items de départ en fonction de la race est spécifiée dans ton fichier personnel.

Étape 17: Le marchand

Le joueur peut se rendre chez un marchand pour acheter des items supplémentaires. Le marchand propose une liste d'items que le joueur peut acheter, chacun ayant un prix en pièces d'or et un niveau requis pour l'achat.

Le joueur peut acheter des items tant qu'il a assez de pièces d'or et de la place dans son inventaire (max 5 items).

Astuce: une boucle `while` semble une bonne idée... ton utilisateur reste dans la boutique tant qu'il n'a pas appuyé sur 0 **ET** qu'il lui reste de l'or **ET** qu'il lui reste de la place dans son inventaire.

Il peut quitter la boutique en tapant 0.

Exemples d'items disponibles chez le marchand :

- Potion de soin (niveau 1, 10 pièces d'or)
- Épée longue (niveau 2, 20 pièces d'or)

- Armure de cuir (niveau 1, 15 pièces d'or)

Quand le joueur a acheté un item, ce dernier disparaît de la boutique et il faut **décompter le prix d'achat** du nombre de pièces d'or disponibles.

Après chaque achat, il faut afficher l'item acheté, l'inventaire et le nombre de pièces d'or restantes.

Exemple :

```
Votre inventaire :
-    Épée courte
-    Bouclier
-    Potion de soin

Il vous reste 10 pièces d'or

[+ afficher les items disponibles]
```

Implémentation de BASE

Tu peux afficher toute la liste de ce qu'il y a chez le marchand. Il suffit alors de vérifier au moment du choix si ton personnage possède le niveau et le nombre de pièces d'or nécessaires pour réaliser cet achat.

Implémentation BONUS

Si tu préfères, tu peux réaliser cette version du marchand. Elle est un peu plus compliquée que l'autre.

Dans ce cas, il ne faut afficher que les éléments que ton personnage peut acheter en fonction de son niveau et du nombre de pièces d'or qu'il lui reste.

Crée une liste intermédiaire qui contient uniquement les items que le joueur peut se procurer (sur base du nombre de pièces d'or et de son niveau). Ajoute la position de l'item dans la liste originale. Il sera ainsi plus facile par la suite pour supprimer cet item de la boutique.

Exemple si ton joueur de niveauc 1 possède 12 pièces d'or, il ne peut choisir que la potion de soin et l'armure de cuir.

```
items_dispo: [
  {
    "nom": "Potion de soin",
    "niveau": 1,
    "prix": 10,
    "index": 0
  },
  {
    "nom": "Armure de cuir",
    "niveau": 1,
    "prix": 15
    "index": 2
  }
]
```

Etape 18: Sauvegarde en JSON

Sauvegarde les données de ton personnage dans un fichier json (extension `.json`).

Nom du fichier: `[pseudo].json`

Tu dois sauvegarder:

- nom
- race
- classe
- les 6 caractéristiques
- pv
- xp
- niveau
- inventaire
- pièces d'or
- 🔥 tests

UPDATE: Ajoute également les tests de caractéristiques à ton fichier. Vu que les tests de caractéristiques sont déjà stockés dans un tableau d'objets, il est très facile de les ajouter ici.

Pour ce faire, créer un objet dictionnaire avec les données adéquates.

Voici le code qui te permet de sauvegarder un objet dans un fichier json:

```python
import json

# Création du dictionnaire du personnage
personnage = {}

# Sauvegarde dans un fichier JSON
with open(f"{pseudo}.json", "w", encoding="utf-8") as f:
    json.dump(personnage, f, indent=2, ensure_ascii=False)
```

La structure du fichier DOIT correspondre à ceci:

```json
{
  "pseudo": "Thorgal",
  "race": "Nain",
  "classe": "Guerrier",
  "caracteristiques": {
    "Force": 14,
    "Sagesse": 12,
    "Intelligence": 10,
    "Dextérité": 13,
    "Constitution": 15,
    "Charisme": 11
  },
  "pv": 28,
  "xp": 350,
  "pieces": 75,
  "inventaire": [
    "Hache",
    "Potion de soin",
    "Anneau magique"
  ],
  "tests": [
    {
      "intitule": "Identifier une créature magique",
      "caracteristique_testee": "Dexterite",
      "degre_difficulte": 19,
      "xp": 350,
      "pv": 0
    },
    {
      "intitule": "Repérer un détail caché",
      "caracteristique_testee": "Force",
      "degre_difficulte": 18,
      "xp": 350,
      "pv": 0
    },
    {
      "intitule": "Nager à contre-courant",
      "caracteristique_testee": "Force",
      "degre_difficulte": 12,
      "xp": 350,
      "pv": 4
    },
    {
      "intitule": "Effectuer une course rapide",
      "caracteristique_testee": "Dexterite",
      "degre_difficulte": 12,
      "xp": 350,
      "pv": 0
    },
    {
      "intitule": "Décrypter un code secret",
      "caracteristique_testee": "Intelligence",
      "degre_difficulte": 18,
      "xp": 350,
      "pv": 0
```

```
    },
    {
      "intitule": "Réparer une machine",
      "caracteristique_testee": "Force",
      "degre_difficulte": 11,
      "xp": 350,
      "pv": 0
    },
    {
      "intitule": "Analyser une situation",
      "caracteristique_testee": "Sagesse",
      "degre_difficulte": 18,
      "xp": 350,
      "pv": 0
    },
    {
      "intitule": "Réparer une machine",
      "caracteristique_testee": "Intelligence",
      "degre_difficulte": 20,
      "xp": 350,
      "pv": 0
    },
    {
      "intitule": "Endurer des conditions extrêmes",
      "caracteristique_testee": "Constitution",
      "degre_difficulte": 12,
      "xp": 350,
      "pv": 8
    },
    {
      "intitule": "Cuisiner un plat délicat",
      "caracteristique_testee": "Sagesse",
      "degre_difficulte": 17,
      "xp": 350,
      "pv": 1
    }
  ]
}
```