Mini-projet n°1, N.S.I

Jeu de dés

Du 26 Novembre 2024 au 16 Décembre 2024

PARTICIPANTS

SIMON Denevan, CHAPMAN Matt

PRÉSENTATION

On souhaite créer un jeu en langage Python. Voici la règle: 2 joueurs lancent trois dés et doivent créer le plus grand nombre à partir des 3 chiffres obtenus. Le joueur avec le plus grand nombre gagne.

RÉPARTITION DES TÂCHES

Voici la répartition des fonctions à faire:

- **Denevan**
  - **nombre_aleatoire(n)**
  - **couleur_aleatoire()**
  - **tracer_carre(x, y, longueur)**
  - **tracer_point(x, y, longueur)**
  - **afficher_diagonale_1(x, y, longueur)**
  - **afficher_diagonale_2(x, y, longueur)**
- **Matt**
  - **afficher_message(x, y, texte)**
  - **comparer_chiffre(chiffre1, chiffre2, chiffre3)**
  - **choisir_face_a_afficher(x, y, lance, longueur)**
  - **afficher_un(x, y, longueur)**
  - **afficher_horizontale_milieu(x, y, longueur)**

Le lancer_jeu() a été fait ensemble.

LES FONCTIONS

afficher_message(x, y, texte)

```
def afficher_message(x, y, texte): # Matt
    """
    Procédure affichant le message 'texte' aux coordonnés x,y.
    """
    ## Ecrivez ici le code de la fonction
    goto(x, y)
    write(texte, align="center", font=("Georgia", 27, "normal"))
```

Cette fonction consiste à écrire un message. Au début, j'avais mis *align=center* sans les " ". J'ai donc eu cette erreur: *NameError: name 'center' is not defined.*

J'ai donc cherché et il comptait *center* comme une variable.

comparer_chiffre(chiffre1, chiffre2, chiffre3)

```
62      combi1 = chiffre1 * 100 + chiffre2 * 10 + chiffre3
63      combi2 = chiffre1 * 100 + chiffre3 * 10 + chiffre2
64      combi3 = chiffre2 * 100 + chiffre1 * 10 + chiffre3
65      combi4 = chiffre2 * 100 + chiffre3 * 10 + chiffre1
66      combi5 = chiffre3 * 100 + chiffre1 * 10 + chiffre2
67      combi6 = chiffre3 * 100 + chiffre2 * 10 + chiffre1
68
69
70
71      if combi1 >= combi2 and combi1 >= combi3 and \
72         combi1 >= combi4 and combi1 >= combi5 and \
73         combi1 >= combi6:
74          return combi1
75      elif combi2 >= combi3 and combi2 >= combi4 and \
76           combi2 >= combi5 and combi2 >= combi6:
77          return combi2
78      elif combi3 >= combi4 and combi3 >= combi5 and \
79           combi3 >= combi6:
80          return combi3
81      elif combi4 >= combi5 and combi4 >= combi6:
82          return combi4
83      elif combi5 >= combi6:
84          return combi5
85      else:
86          return combi6
```

Cette fonction reçoit 3 chiffres, il doit la plus grande combinaison.

De la ligne 62 à 67, la fonction teste toutes les combinaisons possibles. De la ligne 71 à 86, on teste toutes les combinaisons pour trouver la plus grande. Le \ sert à continuer le code sur plusieurs lignes, pour que le code soit plus clair.

afficher_un(x, y, longueur)

```
def afficher_un(x, y, longueur): # Matt
    """
    Procédure affichant le point milieu d'un dé, dont la longueur d'un
    coté est 'longueur'. Ce tracé s'effecue à partir des coordonnées x,y du
    point inférieur gauche de ce dé.
    """
    ## Ecrivez ici le code de la fonction
    penup()
    tracer_point(x+(longueur/2),y+(longueur/2), longueur)
```

La fonction *afficher_un* doit afficher le chiffre un d'un dés, soit un seul point central. Il va donc au milieu de la longueur du futur cube et place un point.

afficher_horizontal_milieu(x, y, longueur)

```
penup()
tracer_point(x+(1/4*longueur), y+(1/2*longueur), longueur)
tracer_point(x+(3/4*longueur), y+(1/2*longueur), longueur)
```

La fonction doit faire les deux points du milieu du chiffre 6. 2 points sont alors faits au centre côté gauche et centre côté droit.

choisir_face_a_afficher(x, y, lance, longueur)

```
    if lance == 1:
        tracer_carre(x, y, longueur)
        afficher_un(x, y, longueur)

    elif lance == 2:
        tracer_carre(x, y, longueur)
        afficher_diagonale_1(x, y, longueur)

    elif lance == 3:
        tracer_carre(x, y, longueur)
        afficher_un(x, y, longueur)
        afficher_diagonale_1(x, y, longueur)

    elif lance == 4:
        tracer_carre(x, y, longueur)
        afficher_diagonale_1(x, y, longueur)
        afficher_diagonale_2(x, y, longueur)

    elif lance == 5:
        tracer_carre(x, y, longueur)
        afficher_diagonale_1(x, y, longueur)
        afficher_diagonale_2(x, y, longueur)
        afficher_un(x, y, longueur)

    elif lance == 6:
        tracer_carre(x, y, longueur)
        afficher_diagonale_1(x, y, longueur)
        afficher_diagonale_2(x, y, longueur)
        afficher_horizontale_milieu(x, y, longueur)
```

La fonction *choisir_face_a_afficher* doit afficher une face donné via la variable *lance*. Si le lancé est un, il fait le carré et *affiche_un*. Si le lancé est deux, on fait le carré et une diagonale. Si trois, on fait une diagonale et un point central. Si c'est quatre, on fait deux diagonales. Si cinq, on fait exactement le même que quatre mais on ajoute un point central. Enfin, pour le six, on fait les deux diagonales et on ajoute les deux points des côtés au centre en plus.

lancer_jeu()

Cette fonction finale permet de lancer le jeu, on y retrouve toutes les fonctions ci-dessus, plusieurs demandes tels que les noms des joueurs, rejouer ou non. On y retrouve aussi une ligne de commande pour ouvrir directement en grande fenêtre.

BIBLIOGRAPHIE/SITOGRAPHIE

Documentation turtle:

https://docs.python.org/fr/3/library/turtle.html

Pour paramétrer la fenêtre turtle:

https://zestedesavoir.com/tutoriels/944/a-la-decouverte-de-turtle/configurer-la-fenetre/