1. Introduction à Python

Qu'est-ce que Python ?

- Python est un langage de programmation de haut niveau, interprété et polyvalent.
- Il est célèbre pour sa syntaxe claire et sa facilité d'apprentissage.

Premiers pas avec Python

- Python utilise des "scripts" pour exécuter des commandes.
- Un script Python se termine généralement par `.py`.

Exemple de Code : Hello World

```
# Ceci est un commentaire - il n'est pas exécuté par Python
# Imprime un message à l'écran
print("Bonjour le monde !")
```

2. Les Bases de Python

Variables en Python

- Une variable est comme une boîte où vous pouvez stocker des informations.
- Exemple : `nombre = 5` stocke la valeur 5 dans la variable `nombre`.

Utiliser `input()` pour Saisir des Données

- `input()` permet à l'utilisateur de saisir des données.
- Exemple : `nom = input("Comment vous appelez-vous ? ")`

Exemple de Code : Salutation Personnalisée

```
# Demande le nom de l'utilisateur
nom = input("Comment vous appelez-vous ? ")

# Affiche une salutation
print("Bonjour, " + nom + "!")
```

3. Structures Conditionnelles

Utiliser `if-else` en Python

- `if-else` permet de prendre des décisions en fonction des conditions.
- Exemple : Exécuter un bloc de code seulement si une condition est vraie.

Exemple de Code : Pair ou Impair

```
# Demande un nombre à l'utilisateur
nombre = int(input("Entrez un nombre : "))
# Vérifie si le nombre est pair ou impair
if nombre % 2 == 0:
    print("Le nombre est pair.")
else:
    print("Le nombre est impair.")
```

4. Boucles en Python

Boucles `for`

- Une boucle `for` répète une action un certain nombre de fois.
- Très utile pour parcourir des séquences (comme des listes ou des chaînes).

Exemple de Code : Afficher les Nombres de 1 à 5

```
# Utilise une boucle for pour afficher les nombres de 1 à 5
for i in range(1, 6):
    print(i)
```

Exemple de Code : Table de Multiplication

```
# Demande un nombre à l'utilisateur
nombre = int(input("Entrez un nombre : "))
# Affiche la table de multiplication pour ce nombre
for i in range(1, 11):
    print(nombre, "x", i, "=", nombre * i)
```

5. Activités Python et Mathématiques

Calculer la superficie d'un cercle :

```
# Programme pour calculer la superficie d'un cercle
# Demander à l'utilisateur de saisir le rayon du cercle
rayon = float(input("Entrez le rayon du cercle : "))
# Calculer la superficie du cercle en utilisant la formule : pi * r^2
superficie = 3.14 * (rayon ** 2)
# Afficher le résultat
print("La superficie du cercle est :", superficie)
```

Donner le maximum de deux nombres :

```
# Programme pour trouver le maximum de deux nombres
# Demander à l'utilisateur de saisir deux nombres
nombre1 = float(input("Entrez le premier nombre : "))
nombre2 = float(input("Entrez le deuxième nombre : "))
# Trouver le maximum des deux nombres
maximum = max(nombre1, nombre2)
# Afficher le résultat
print("Le maximum des deux nombres est :", maximum)
```

Vérifier si un nombre est pair :

```
# Programme pour vérifier si un nombre est pair
# Demander à l'utilisateur de saisir un nombre
nombre = int(input("Entrez un nombre : "))
# Vérifier si le nombre est pair en utilisant l'opérateur modulo (%)
if nombre % 2 == 0:
   print("Le nombre est pair.")
else:
   print("Le nombre est impair.")
```

Addition de deux nombres :

```
# Programme pour additionner deux nombres
# Demander à l'utilisateur de saisir deux nombres
nombre1 = float(input("Entrez le premier nombre : "))
nombre2 = float(input("Entrez le deuxième nombre : "))
# Additionner les deux nombres
resultat = nombre1 + nombre2
# Afficher le résultat
print("La somme des deux nombres est :", resultat)
```

Calcul de la moyenne de trois nombres :

```
# Programme pour calculer la moyenne de trois nombres
# Demander à l'utilisateur de saisir trois nombres
nombre1 = float(input("Entrez le premier nombre : "))
nombre2 = float(input("Entrez le deuxième nombre : "))
nombre3 = float(input("Entrez le troisième nombre : "))
# Calculer la moyenne des trois nombres
moyenne = (nombre1 + nombre2 + nombre3) / 3
# Afficher le résultat
print("La moyenne des trois nombres est :", moyenne)
```

Vérifier si un nombre est positif, négatif ou nul :

```
# Programme pour vérifier si un nombre est positif, négatif ou nul
# Demander à l'utilisateur de saisir un nombre
nombre = float(input("Entrez un nombre : "))
# Vérifier le signe du nombre
if nombre > 0:
    print("Le nombre est positif.")
elif nombre < 0:
```

```
    print("Le nombre est négatif.")
else:
    print("Le nombre est nul.")
```

6. Introduction à Turtle

Qu'est-ce que Turtle ?

- Turtle est une bibliothèque graphique intégrée à Python, utilisée pour créer des dessins et des animations simples.

Configurer Turtle

- Turtle est inclus dans l'installation standard de Python.
- Pour l'utiliser, vous devez simplement l'importer dans votre script.

Exemple de Code : Votre Premier Dessin

```
import turtle
# Crée un nouvel objet turtle
t = turtle.Turtle()
# Déplace le turtle pour dessiner
t.forward(100)  # Avance de 100 pixels
t.left(90)      # Tourne de 90 degrés
t.forward(100)
# Termine le dessin
turtle.done()
```

7. Dessiner avec Turtle

Commandes de Base

- `forward(x)` : avance de x pixels.
- `backward(x)` : recule de x pixels.
- `right(angle)` : tourne vers la droite de angle degrés.
- `left(angle)` : tourne vers la gauche de angle degrés.

Exemple de Code : Dessiner un Carré

```
import turtle
t = turtle.Turtle()
# Répète 4 fois pour dessiner un carré
for _ in range(4):
    t.forward(100)  # Longueur du côté du carré
    t.left(90)      # Angle pour tourner
turtle.done()
```

Projets Plus Avancés avec Turtle : Utiliser des Boucles

- Les boucles `for` sont utiles pour répéter des actions.

Exemple de Code : Dessiner un Cercle

```
import turtle
t = turtle.Turtle()

# Utiliser une boucle pour dessiner un cercle
for _ in range(360):
    t.forward(1)
    t.left(1)
turtle.done()
```

Tracer des polygones avec une fonction

```
import turtle
def polygonne(couleur, cotes, taille):
  turtle.fillcolor(couleur)  # Définit la couleur de remplissage
  turtle.begin_fill()      # Commence à remplir la couleur
  for _ in range(cotes):
    turtle.forward(taille)  # Avance de la taille spécifiée
    turtle.left(360 / cotes) # Tourne pour former le polygone
  turtle.end_fill()       # Termine le remplissage de la couleur
```

```
# Initialisation de Turtle
window = turtle.Screen()  # Crée une fenêtre pour Turtle
turtle.speed(3)         # Réglage de la vitesse de dessin
# Exemple d'utilisation de la fonction polygonne
polygonne("blue", 5, 100)  # Dessine un pentagone bleu
window.mainloop()  # Garde la fenêtre ouverte
```

Dans ce programme, la fonction polygonne est définie pour créer un polygone selon les spécifications données. Vous pouvez appeler cette fonction avec différentes valeurs pour créer divers polygones. Par exemple, polygonne("rouge", 4, 80) dessinerait un carré rouge.

8. Commandes de Mouvement

- **`forward(distance)` ou `fd(distance)`** : Avance le turtle de 'distance' unités.
- **`backward(distance)` ou `bk(distance)`** : Recule le turtle de 'distance' unités.
- **`right(angle)` ou `rt(angle)`** : Tourne le turtle à droite de 'angle' degrés.
- **`left(angle)` ou `lt(angle)`** : Tourne le turtle à gauche de 'angle' degrés.
- **`goto(x, y)`** : Déplace le turtle à la position spécifiée.
- **`setx(x)`** : Déplace le turtle à la coordonnée x.
- **`sety(y)`** : Déplace le turtle à la coordonnée y.
- **`circle(radius)`** : Dessine un cercle avec le rayon spécifié.
- **`dot(size=None, color=None)`** : Dessine un point.

9. Commandes de Contrôle

- **`speed(speed)`** : Définit la vitesse de mouvement du turtle (1-10).
- **`penup()` ou `pu()`** : Lève le stylo, aucun dessin quand le turtle se déplace.
- **`pendown()` ou `pd()`** : Pose le stylo, dessine quand le turtle se déplace.

10. Commandes de Stylo et Couleur

- **`color(color)`** : Change la couleur du stylo.
- **`begin_fill()`** : Commence à remplir la forme dessinée.
- **`end_fill()`** : Termine le remplissage de la forme.
- **`fillcolor(color)`** : Change la couleur de remplissage.

11. Commandes de Fenêtre

- **`title(title_string)`** : Donne un titre à la fenêtre Turtle.
- **`bgcolor(color)`** : Change la couleur d'arrière-plan de la fenêtre.
- **`clear()`** : Efface le dessin du turtle, mais garde la fenêtre ouverte.
- **`reset()`** : Réinitialise le turtle à sa position et état d'origine.
- **`bye()`** : Ferme la fenêtre Turtle.

12. Couleurs Courantes

- Vous pouvez spécifier des couleurs en utilisant des noms (comme `"red"`, `"green"`, `"blue"`), ou en utilisant des codes hexadécimaux (comme `"#FF0000"` pour rouge).
- Couleurs de base : `"red"`, `"green"`, `"blue"`, `"yellow"`, `"black"`, `"white"`, `"pink"`, `"orange"`, `"purple"`, `"grey"`

Exemple de Code Utilisant Turtle

```
import turtle
turtle.speed(1)  # Vitesse la plus lente
turtle.color("blue")
turtle.forward(100)
turtle.right(90)
turtle.color("green")
turtle.forward(100)
turtle.done()
```

Conclusion

- Vous avez maintenant une base en Python et Turtle.
- Expérimentez avec les commandes pour créer vos propres dessins !
- Faite une recherche sur Idder MOUTIA
- Faite une recherche sur l’ecole 1337