En Route vers Python : Un Cours STEM Amusant pour les Futurs Codeurs

Par : Larbi OUIYZME

Qu'est-ce que STEM ?

STEM est un acronyme qui signifie Science, Technology, Engineering, and Mathematics. Les cours STEM sont conçus pour enseigner et intégrer ces disciplines de manière interdisciplinaire, encourageant les élèves à développer des compétences dans ces domaines cruciaux. L'introduction à Python dans un contexte STEM vise à familiariser les élèves avec la programmation informatique tout en mettant l'accent sur ces disciplines fondamentales. En outre, ce cours abordera les concepts mathématiques, la programmation des robots, ainsi que le développement de jeux.

Qu'est-ce que Python ?

Python est un langage de programmation amusant et puissant que nous allons explorer ensemble. Aujourd'hui, nous allons découvrir quelques notions de base qui vous aideront à démarrer.

Python permet de donner des instructions à l'ordinateur pour effectuer différentes tâches. Il est utilisé dans de nombreux domaines tels que le développement de logiciels, la science des données, l'intelligence artificielle, la robotique et même pour créer des jeux.

Syntaxe Python

La syntaxe, c'est la façon dont on écrit les instructions en Python. Le langage Python est conçu pour être facile à lire et à écrire.

En Python, le nommage des variables suit certaines règles et est sensible à la casse. Voici quelques principes à respecter :

1. Nommage des Variables :
   - Les noms de variables doivent commencer par une lettre (a-z, A-Z) ou un souligné _.
   - Les caractères suivants dans le nom de la variable peuvent être des lettres, des chiffres (0-9) ou des soulignés.
   - Évitez d'utiliser des mots réservés par Python comme noms de variables (par exemple, if, else, for, etc.).
2. Sensibilité à la Casse :

- Python est sensible à la casse, ce qui signifie que les variables maVariable et MAvariable sont considérées comme deux variables distinctes.
- Soyez cohérent dans l'utilisation de la casse pour garantir la clarté et la cohérence du code.

Exemple de bonnes pratiques de nommage :

```
# Noms significatifs et explicites
nom_utilisateur = "JohnDoe"
age_utilisateur = 25
# Utilisation de la convention Snake Case pour des noms composés
ma_variable = 10
total_elements = 50
# Éviter les noms peu explicites ou abrégés
x = 5  # Éviter ce genre de noms, sauf dans des contextes très limités
```

Commentaires en Python :

Les commentaires sont des annotations dans le code source qui ne sont pas exécutées lors de l'exécution du programme. En Python, les commentaires sont créés en utilisant le caractère #.

Exemple :

```
# Ceci est un commentaire sur une seule ligne

"""
Ceci est un commentaire
sur plusieurs lignes
"""
```

Affichage de messages avec print

Dans Python, nous pouvons demander à l'ordinateur d'afficher des messages à l'écran. Utilisons la commande print pour dire quelque chose à notre programme.

```
print("Bonjour les amis !")
```

En exécutant ce code, vous verrez la phrase "Bonjour les amis !" s'afficher à l'écran.

Demander à l'utilisateur avec input

Maintenant, parlons de communication. Comment demandons-nous quelque chose à l'utilisateur ? Utilisons input pour obtenir une réponse de l'utilisateur.

```
nom = input("Comment t'appelles-tu ? ")
print("Salut", nom, "!")
```

Ici, le programme demandera votre nom, puis il l'affichera avec un accueillant "Salut".

N.B :

En Python, lorsque vous utilisez la fonction input(), elle traite automatiquement l'entrée comme une chaîne de caractères (string). Pour convertir cette entrée en nombre entier (int) ou nombre décimal (float), vous pouvez utiliser les fonctions int() et float().

```
# Demander à l'utilisateur d'entrer un nombre décimal
rayon = float(input("Entrez un nombre décimal : "))
# Demander à l'utilisateur d'entrer un nombre entier
bonbons = int(input("Entrez un nombre décimal : "))
# Afficher le résultat
print("Le nombre décimal est :", rayon)
print("Le nombre entierl est :", bonbons)
```

Découvrir le type de données avec type

Python peut manipuler différents types de données, comme des nombres ou des mots. Pour savoir de quel type est une donnée, nous utilisons type.

```
age = 10
nom = "Alice"
print(type(age))  # Affiche <class 'int'>
print(type(nom))  # Affiche <class 'str'>
```

En utilisant type, nous pouvons comprendre le type de données avec lequel nous travaillons.

Mesurer la longueur avec len

Parfois, nous voulons savoir combien de lettres ou de chiffres il y a dans une variable ou autre.
len nous aide à mesurer la longueur.

```
mot = "Python"
longueur = len(mot)
```

print("Le mot", mot, "a", longueur, "lettres.")

Ici, le programme dira combien de lettres il y a dans le mot "Python".

Variables

Une variable est un espace dans la mémoire de l'ordinateur où l'on peut stocker des données. On peut donner n'importe quel nom à une variable. Par exemple :

# Déclarer une variable
nom = "Python"

Types de Variables en Python

En Python, il existe plusieurs types de variables pour stocker différentes sortes de données. Voici quelques-uns des types de variables les plus couramment utilisés :

Entier (int) : Stocke des nombres entiers, par exemple, 5, -10, 100.
age = 10

Flottant (float) : Stocke des nombres décimaux, par exemple, 3.14, -0.5.
pi = 3.14

Chaîne de caractères (str) : Stocke du texte, par exemple, "Python", 'Bonjour'.
prenom = "Alice"

Booléen (bool) : Stocke soit True (Vrai) soit False (Faux).
est_vrai = True

Opérations Arithmétiques

Les opérations arithmétiques sont utilisées pour effectuer des calculs mathématiques. Voici quelques-unes des opérations arithmétiques de base :

Addition (+) : Additionne deux nombres.
somme = 5 + 3

Soustraction (-) : Soustrait un nombre d'un autre.
difference = 7 - 2

Multiplication (*) : Multiplie deux nombres.
produit = 4 * 6

Division (/) : Divise un nombre par un autre.
quotient = 10 / 2

Division entiere(//): Divise un nombre par un autre et le resultat est un nombre entier de type int.
quotient = 10 // 3 # Résultat est 3

Modulo (%) : Donne le reste de la division.
reste = 15 % 4 # Résultat est 3

Opérations Logiques et de Comparaison

Les opérations logiques et de comparaison sont utilisées pour évaluer des conditions logiques. Voici quelques-unes d'entre elles :
Opérations de Comparaison :

Égalité (= =) : Vérifie si deux valeurs sont égales.
resultat = (5 == 5) # True

Inégalité (!=) : Vérifie si deux valeurs ne sont pas égales.
resultat = (5 != 3) # True

Boucle for

La boucle for permet de répéter une série d'instructions un certain nombre de fois. Par exemple :

```
# Afficher les nombres de 1 à 5
for i in range(1, 6):
    print(i)
```

Boucle while

La boucle while permet de répéter des instructions tant qu'une condition est vraie. Par exemple :

```
# Afficher les carrés des nombres de 1 à 5
nombre = 1
while nombre <= 5:
    print(nombre ** 2)
    nombre += 1
```

Structure conditionnelle if, else

La structure conditionnelle if permet d'exécuter des instructions uniquement si une condition est vraie. La structure else permet d'exécuter des instructions alternatives si la condition n'est pas vraie. Par exemple :

```
# Vérifier si un nombre est pair ou impair
nombre = int(input("Entrez un nombre : "))
if nombre % 2 == 0:
    print("Le nombre est pair.")
else:
    print("Le nombre est impair.")
```

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.")
```

Forme condensée d'écriture pour effectuer une opération arithmétique et mise à jour de la variable en une seule étape

Les opérateurs +=, -=, *=, et /= offrent une manière concise et efficace d'effectuer une opération arithmétique et de mettre à jour une variable en une seule étape. Ces opérateurs sont couramment utilisés en programmation pour simplifier le code et améliorer la lisibilité.

Voici une brève introduction avec des exemples pour chacun de ces opérateurs :

+= (Addition et assignation) : L'opérateur += permet d'ajouter la valeur à droite de l'opérateur à la variable à gauche, puis d'assigner le résultat à cette variable. Exemple :

```
x = 5
x += 3  # Équivalent à x = x + 3
print(x)  # Affiche 8
```

-= (Soustraction et assignation) : L'opérateur -= effectue une soustraction entre la valeur à droite et la variable à gauche, puis assigne le résultat à la variable. Exemple :

```
y = 10
y -= 4  # Équivalent à y = y - 4
print(y)  # Affiche 6
```

***= (Multiplication et assignation) :** L'opérateur *= multiplie la variable à gauche par la valeur à droite, puis assigne le produit à la variable. Exemple :

```
z = 3
z *= 2  # Équivalent à z = z * 2
print(z)  # Affiche 6
```

/= (Division et assignation) : L'opérateur /= divise la variable à gauche par la valeur à droite, puis assigne le quotient à la variable. Exemple :

```
w = 8
w /= 4  # Équivalent à w = w / 4
print(w)  # Affiche 2.0
```

Calculer le factoriel d'un nombre :

```
# Programme pour calculer le factoriel d'un nombre
# Demander à l'utilisateur de saisir un nombre entier positif
n = int(input("Entrez un nombre entier positif : "))
# Initialiser le résultat à 1
resultat = 1
# Calculer le factoriel en utilisant une boucle
for i in range(1, n + 1):
    resultat *= i
# Afficher le résultat
print(f"Le factoriel de {n} est :", resultat)
```