Travaux pratiques - Révision des classes Python

# Objectifs

Partie 1 : Lancer la DEVASC VM

Partie 2 : Examen des fonctions, des méthodes et des classes

Partie 3 : Définir une fonction

Partie 4 : Définir une classe avec des méthodes

Partie 5 : Passez en revue le script circleClass.py

# Contexte/scénario

Dans ce TP, vous passez en revue les méthodes, les fonctions et les classes Python. Vous créez ensuite une classe et instanciez plusieurs fois avec des valeurs différentes. Enfin, vous passez en revue l'exemple de classe **Circle** utilisé dans le cours.

# Ressources requises

* 1 PC avec système d'exploitation de votre choix
* Boîte virtuelle ou VMWare
* Machine virtuelle DEVASC

# Instructions

## Lancer la machine virtuelle DEVASC

Si vous n'avez pas encore terminé **TP - Installer l'environnement de laboratoire de machines virtuelles**, faites-le maintenant. Si vous avez déjà terminé ce TP, lancez la machine virtuelle DEVASC maintenant.

## Examiner les fonctions, les méthodes et les classes

Dans cette partie, vous passez en revue la différence entre les fonctions et les méthodes. Vous passez également en revue la structure de base d'une classe.

### Qu'est-ce qu'une fonction ?

Comme un examen rapide, rappelez-vous qu'une fonction est un bloc de code défini indépendamment qui est appelé par son nom. Dans l'exemple suivant, la fonction appelée **FunctionName** est définie puis appelée. Notez qu'il s'agit d'un bloc de code indépendant. Il n'est encapsulé dans aucun autre code.

# Define the function

def functionName:

...blocks of code...

# Call the function

# Call the function

### Qu'est-ce qu'une méthode ?

Une méthode, cependant, ne peut pas être appelée par elle-même. Elle dépend de l'objet dans lequel elle est définie. Dans l'exemple suivant, la classe **ClassName** est déclarée et trois méthodes sont définies. La classe est instanciée, puis chaque méthode de la classe est appelée.

**Remarque** : Ce pseudo-code n'affiche pas explicitement la méthode du constructeur de classe  **\_\_init\_\_** avec la variable **self . Cette méthode spéciale est examinée ci-dessous.**

# Define the class

class className

# Define a method

def method1Name

...blocks of code

# Define another method

def method2Name

...blocks of code

# Define yet another method

def method3Name

...blocks of code

# Instantiate the class

myClass = className()

# Call the instantiation and associated methods

myClass.method1Name()

myClass.method2Name()

myClass.method3Name()

## Définir une fonction

Dans cette partie, vous allez définir une fonction avec des arguments, puis appeler la fonction.

* + - 1. Ouvrez un nouveau fichier texte et enregistrez-le sous **myCity.py** dans votre répertoire **~/labs/devnet-src/python**.
      2. Définissez la fonction **myCity** avec l'argument **city** pour le nom de la ville. Lorsque la fonction est appelée avec un nom de ville spécifié, elle imprime une instruction qui inclut le nom de la ville.

def myCity(city):

print("I live in " + city + ".")

* + - 1. Appelez la fonction **myCity** en lui transmettant des valeurs différentes pour **ville**, comme indiqué dans les exemples suivants.

myCity("Austin")

myCity("Tokyo")

myCity("Salzburg")

* + - 1. Enregistrez et exécutez le fichier **myCity.py**. Vous devriez obtenir la sortie suivante.

devasc@labvm:~/labs/devnet-src/python$ **python3 myCity.py**

I live in Austin.

I live in Tokyo.

I live in Salzburg.

devasc@labvm:~/labs/devnet-src/python$

## Définir une classe avec des méthodes

Dans cette partie, vous allez définir une classe, utiliser la fonction \_\_init\_\_ () pour définir une méthode pour la classe, puis créer des instances de la classe.

### Définissez puis instanciez une classe avec la méthode \_\_init\_\_ ().

Une classe Python est utilisée pour créer des objets qui ont des propriétés et des méthodes. Toutes les classes Python incluent généralement une fonction **\_\_init\_\_ ()** explicitement définie, bien que vous puissiez créer une classe sans en définir une. La fonction **\_\_init\_\_ ()** est toujours **init**iated lorsqu'une classe est instanciée. Instancier une classe crée une copie de la classe qui hérite de toutes les variables et méthodes de classe.

**Remarque** : Bien qu'il soit parfois appelé la fonction **\_\_init\_\_ ()** , il dépend de la classe. Par conséquent, c'est techniquement une méthode.

* + - 1. Ouvrez un nouveau fichier texte et enregistrez-le sous **myLocation.py**.
      2. Définissez une classe portant le nom **Location** et appuyez sur Entrée. Si vous travaillez est VS Code, alors l'éditeur de texte doit automatiquement mettre en retrait quatre espaces.

class Location:

|<-- cursor should now be here

* + - 1. Ensuite, définissez la fonction **\_\_init\_\_ ()**. Par convention, le premier paramètre est appelé **self**. Le paramètre **self** est une référence à l'instance actuelle de la classe elle-même et est utilisé pour accéder aux variables qui appartiennent à la classe entière. La fonction **\_\_init\_\_ ()** est ensuite affectée à toutes les variables dont la classe entière a besoin. Dans l'exemple suivant, définissez une variable de nom et de pays. Appuyez deux fois sur Entrée, puis sur l'espace arrière deux fois vers la marge gauche.

def \_\_init\_\_(self, name, country):

self.name = name

self.country = country

|<-- cursor should now be here

* + - 1. Vous pouvez tester que cette classe est maintenant prête à l'emploi. Instanciez la classe en lui attribuant un nom de votre choix. Ensuite, spécifiez les valeurs pour le **name** et le **country**des variables de classe requises. L'exemple suivant utilise la classe Location pour instancier une classe appelée **loc** avec un **name** et un **country** spécifiés par vous. Utilisez votre nom et votre pays.

loc = Location("Your\_Name", "Your\_Country")

* + - 1. Pour vérifier que la classe **loc** instanciée a maintenant le nom et le pays qui vous sont assignés, ajoutez des instructions d'impression à votre script.

print (loc.name)

imprimer (loc.country)

* + - 1. Pour vérifier que la loc est effectivement une classe, ajoutez l'instruction print suivante qui imprimera le type de données pour **loc**.

print (type (loc))

* + - 1. Enregistrez et exécutez votre script. Vous devriez obtenir la sortie suivante sauf avec votre nom et votre pays fournis.

devasc@labvm:~/labs/devnet-src/python$ **python3 myLocation.py**

Your\_Name

Your\_Country

<class '\_\_main\_\_.Location'>

devasc@labvm:~/labs/devnet-src/python$

### Ajoutez une méthode à la classe Location.

Maintenant, ajoutez une méthode à la classe **Location** qui peut être appelée par un programmeur lorsque la classe est instanciée. Dans cet exemple simple, créez une méthode pour imprimer l'instruction, « Mon nom est [nom] et je vis dans [pays] ».

* + - 1. Supprimez le code qui commence par l'instanciation de la classe **loc**. Votre script **myLocation.py** doit désormais inclure uniquement le code suivant.

class Location:

def \_\_init\_\_(self, name, country):

self.name = name

self.country = country

* + - 1. Avec votre curseur à la fin de la ligne **self.country = country**, appuyez deux fois sur la touche Entrée et arrière une fois.

self.country = country

|<--Your cursor should be here

* + - 1. Définissez un nouvel appel de méthode **myLocation** et attribuez-lui le paramètre **self** afin que la nouvelle méthode puisse accéder aux variables définies dans la fonction **\_\_init\_\_ ()**. Ensuite, définissez une instruction print pour imprimer la chaîne spécifiée ci-dessus.

**Remarque** : La déclaration d'impression doit être sur une ligne.

def myLocation(self):

print("Hi, my name is " + self.name + " and I live in " + self.country + ".")

* + - 1. Appuyez deux fois sur la touche Entrée et deux fois sur l'espace arrière.
      2. Enregistrez et exécutez votre script pour vous assurer qu'il n'y a pas d'erreurs. Vous n'obtiendrez pas encore de sortie.

### Instanciez la classe Location plusieurs fois et appelez la méthode myLocation.

Maintenant que vous avez une classe, vous pouvez l'instancier autant de fois que vous le souhaitez en fournissant des valeurs différentes pour les variables de classe à chaque fois.

* + - 1. Ajoutez le code suivant à votre script **myLocation.py** pour instancier la classe **Location** et appeler la méthode. Vous n'avez pas besoin d'ajouter les commentaires.

# First instantiation of the class Location

loc1 = Location("Tomas", "Portugal")

# Call a method from the instantiated class

loc1.myLocation()

* + - 1. Save and run your script. Vous devriez obtenir la sortie suivante.

devasc@labvm:~/labs/devnet-src/python$ **python3 myLocation.py**

Hi, my name is Tomas and I live in Portugal.

devasc@labvm:~/labs/devnet-src/python$

* + - 1. Ajoutez deux instanciations supplémentaires, puis une quatrième où vous spécifiez le nom et les valeurs de **votre\_loc**.

loc2 = Location("Ying", "China")

loc3 = Location("Amare", "Kenya")

loc2.myLocation()

loc3.myLocation()

your\_loc = Location("Your\_Name", "Your\_Country")

your\_loc.myLocation()

* + - 1. Save and run your script. Vous devriez obtenir la sortie suivante.

devasc@labvm:~/labs/devnet-src/python$ **python3 myLocation.py**

Hi, my name is Tomas and I live in Portugal.

Hi, my name is Ying and I live in China.

Hi, my name is Amare and I live in Kenya.

Hi, my name is Your\_Name and I live in Your\_Country.

devasc@labvm:~/labs/devnet-src/python$

### Passez en revue le script myLocation.py complet.

Si vous avez eu des erreurs avec votre script, consultez l'exemple suivant qui inclut tout le code utilisé dans cette partie.

# Define a class with variables for \*\*name\*\* and \*\*country\*\*.

# Then define a method that belongs to the class. The method’s

# purpose is to print a sentence that uses the variables.

class Location:

def \_\_init\_\_(self, name, country):

self.name = name

self.country = country

def myLocation(self):

print("Hi, my name is " + self.name + " and I live in " + self.country + ".")

# First instantiation of the Location class

loc1 = Location("Tomas", "Portugal")

# Call a method from the instantiated class

loc1.myLocation()

# Three more instantiations and method calls for the Location class

loc2 = Location("Ying", "China")

loc3 = Location("Amare", "Kenya")

loc2.myLocation()

loc3.myLocation()

your\_loc = Location("Your\_Name", "Your\_Country")

your\_loc.myLocation()

## Passez en revue le script circleClass.py

L'exemple du cours montre comment créer une classe qui calcule la circonférence d'un cercle, puis imprimer la valeur calculée. Il y a quelques choses à noter dans ce script.

* La classe comprend trois méthodes, y compris la fonction **\_\_init\_\_ ()**. La fonction **\_\_init\_\_ ()** fournit une méthode pour entrer la valeur du rayon.
* La méthode **circumference** calcule la circonférence et renvoie la valeur la stockant dans la variable **CircumferenceValue**.
* La méthode **printCircumference** imprime une chaîne. Notez que les variables sont castées sous forme de chaînes avec la fonction **str ()**. Sinon, l'instruction print lancerait une erreur car **self.radius** et **myCircumference** ne sont pas des chaînes.
* La classe Circle a instancié trois fois.

# Given a radius value, print the circumference of a circle.

# Formula for a circumference is c = pi \* 2 \* radius

class Circle:

def \_\_init\_\_(self, radius):

self.radius = radius

def circumference(self):

pi = 3.14

circumferenceValue = pi \* self.radius \* 2

return circumferenceValue

def printCircumference(self):

myCircumference = self.circumference()

print ("Circumference of a circle with a radius of " + str(self.radius) + " is " + str(myCircumference))

# First instantiation of the Circle class.

circle1 = Circle(2)

# Call the printCircumference for the instantiated circle1 class.

circle1.printCircumference()

# Two more instantiations and method calls for the Circle class.

circle2 = Circle(5)

circle2.printCircumference()

circle3 = Circle(7)

circle3.printCircumference()

Fin du document