Exercice ML5.2 - Solution :

Voir solution complète dans le fichier MLExo5.2-TENSORFLOW-DIGITS-my-softmax.py

```
# Définir la fonction my_softmax :    Entrée : Les Pré-activations d'une image,   Sortie : les activations de l'image

# Méthode-1
def my_softmax(PreActiv):
    # calcul du dénominateur : somme des exp(z)
    somme = 0
    for z in PreActiv:
        somme = somme + np.exp(z)

    # calcul de exp(z) / somme
    Activation = []
    for z in PreActiv:
        soft = np.exp(z) / somme
        Activation.append(soft)

    return Activation
```

Exercice ML5.2 - Solution :

```
# Méthode-2
def my_softmax(PreActiv):

    # calcul du dénominateur : somme des exp(z)
    exp = np.exp(PreActiv)
    somme = exp.sum()

    # calcul de exp(z) / somme
    Activation = exp / somme

    return Activation


# Obtenir le vecteur des 10 pré-activations
PreActivations = model.predict(X_test)

# Ici la fonction predict recoit un tableau de 360 images (X_test)
# le modèle a donc fait les prédictions de chacune des 360 images du Test Set.
```

Exercice ML5.2 - Solution :

```
# output_layer AVANT my_softmax

print("PreActivations.shape : ", PreActivations.shape)

# output_layer.shape :  (360, 10)
# La fonction predict retourne un tableau (np.array) de 360 PreActivations.
# Chaque PreActivation est un tableau de 10 nombres.

# Regardons la sortie de la 1ère image  (index=0):
print("PreActivations[0] : ", PreActivations[0])

# Créons les predictions en appliquant l'activation sur chaque Preactivation via my_softmax()

predictions = []
for PreActiv in PreActivations:
    Activation = my_softmax(PreActiv)
    predictions.append(Activation)
```

Exercice ML5.2 - Solution :

```
# convertir predictions en array
predictions = np.array(predictions)
print("predictions.shape : ", predictions.shape)

# Regardons la prediction de la 1ère image  (index=0):
print("prediction 1ère image : ", predictions[0])
```