LA MANU ÉCOLE SUPÉRIEURE DES MÉTIERS DU NUMÉRIQUE

Algorithmique

Séance 8 – Fonctions

NOVEI FORMATION

Définition

Définition d'une fonctions

Une fonction est un ensemble d'instructions qui effectue une tâche précise et peut être réutilisée. En python, les fonctions sont définies par le mot clé « **def** » suivi du nom de la fonction et de parenthèses:

```
def add():
        # instructions
```

LA MANU ÉCOLE SUPÉRIEURE DES MÉTIERS DU NUMÉRIQUE

Retour

Retour d'une valeur

Une fonction peut **retourner une valeur** en utilisant le mot-clé « **return** » :

```
def add():
    # instructions
    return value
```

Cette valeur sera récupérée lors de l'appel de la fonction :

```
result = add()
```

Une fonction ne retournant aucune valeur s'appelle une **procédure**.

Si l'appel est stocké dans une variable sa valeur sera « None ».

Retour de plusieurs valeurs

En Python, une fonction peut retourner plusieurs valeurs :

```
def add():
        # instructions
        return value1, value2
```

Les valeurs sont retournées sous forme de tuple :

```
result = add() # result est un tuple
```

Il est possible de dissocier ces valeurs dans plusieurs variables (déballage du tuple) :

```
result1, result2 = add()
```

Arguments

Passage de paramètres

Entre les parenthèses vous pouvez définir des paramètres **:**

```
def add(a, b, c):
      # instructions
```

Passage des arguments:

```
add(1, 2, 4)
```

Arguments nommés

Afin de s'assurer que les valeurs passées à une fonction correspondent bien à un argument, il est possible de nommer les arguments lors de l'appel à la fonction :

```
add(a=1, b=2, c=4)
```

En nommant les arguments, il est possible de passer les valeurs dans n'importe quel ordre :

```
add(c=4, b=2, a=1)
```

Il faudra cependant passer les arguments nommés en dernier:

```
add(2, 4, a=1)
```

Passage d'arguments avec *args et **kwargs (1/2)

La syntaxe *args permet d'indiquer qu'une fonction peut accepter un nombre variable d'arguments. Ces arguments sont intégrés dans un tuple:

```
def somme(*args):
    s = 0
    for n in args:
        s += n
    return s

result = somme(5,9,6,3,7)
```

Passage d'arguments avec *args et **kwargs (2/2)

La syntaxe **kwargs, de la même manière, permet à une fonction de recevoir un nombre variable d'arguments mais cette fois-ci les arguments devront être passés sous la forme d'un dictionnaire :

```
def person(**kwargs):
    for i, j in kwargs.items():
        print(i, j)

person(firstname="Jane", lastname="DOE", age=67)
```

*args et **kwargs pour séparer les données

Les syntaxes *args et **kwargs peuvent être utilisées pour réaliser les opérations inverse de celles présentés ci-dessus, à savoir séparer des données :

```
def somme(a, b, c):
    s = a + b + c
    return s

x = [1, 2, 4]
somme(*x)
```

On, utilisera la syntaxe *args pour séparer les arguments présents dans une liste ou un tuple et la syntaxe **kwargs pour séparer les arguments présents dans un dictionnaire et fournir des arguments nommés à une fonction.

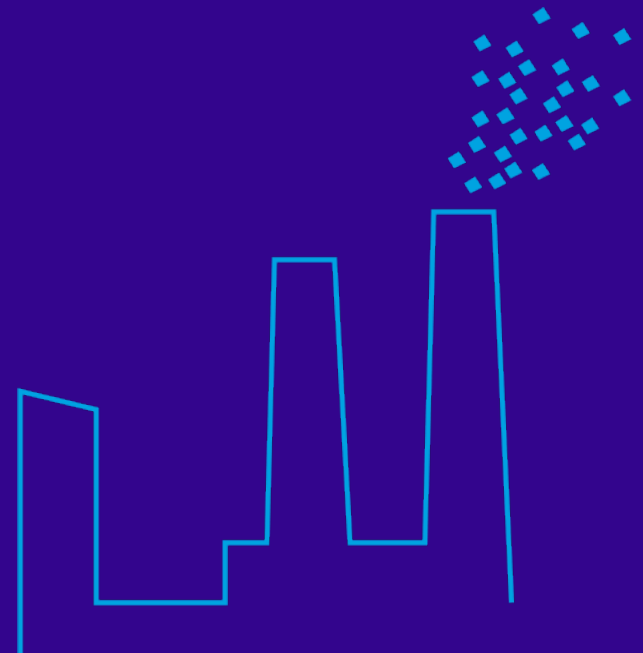
LA MANU ÉCOLE SUPÉRIEURE DES MÉTIERS DU NUMÉRIQUE

NOVEI FORMATION

Rendez-vous sur lamanu.fr

contact@lamanu.fr

09 86 27 17 04



LA MANU ÉCOLE SUPÉRIEURE DES MÉTIERS DU NUMÉRIQUE

NOVEI FORMATION

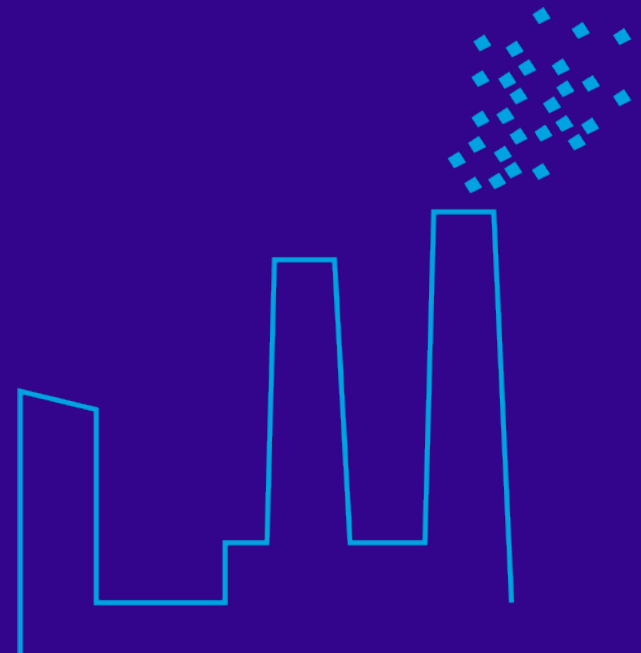
Campus Versailles

contact-versailles@lamanu.fr

https://lamanu.fr

143 rue Yves le Coz, 78000 Versailles

09 86 27 17 04



LA MANU ÉCOLE SUPÉRIEURE DES MÉTIERS DU NUMÉRIQUE

NOVEI FORMATION

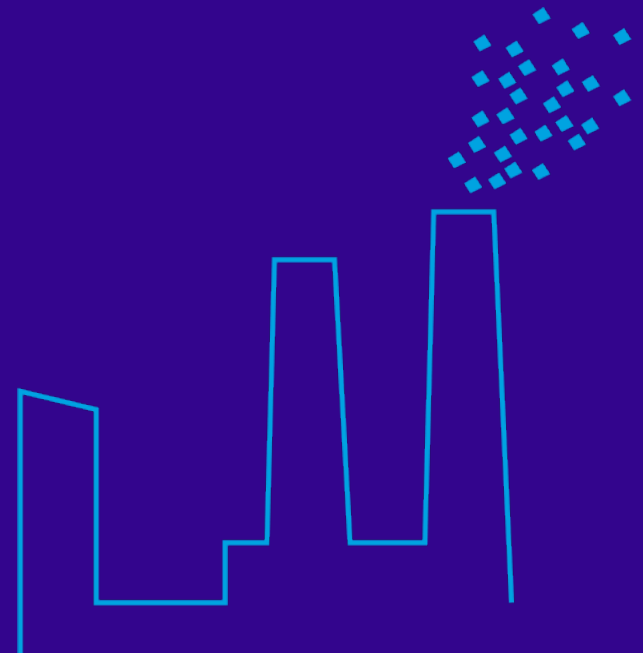
Campus Le Havre

contact-lehavre@lamanu.fr

https://lamanu.fr

10 place Léon Meyer, 76600 Le Havre

09 86 27 17 04



LA MANU ÉCOLE SUPÉRIEURE DES MÉTIERS DU NUMÉRIQUE

NOVEI FORMATION

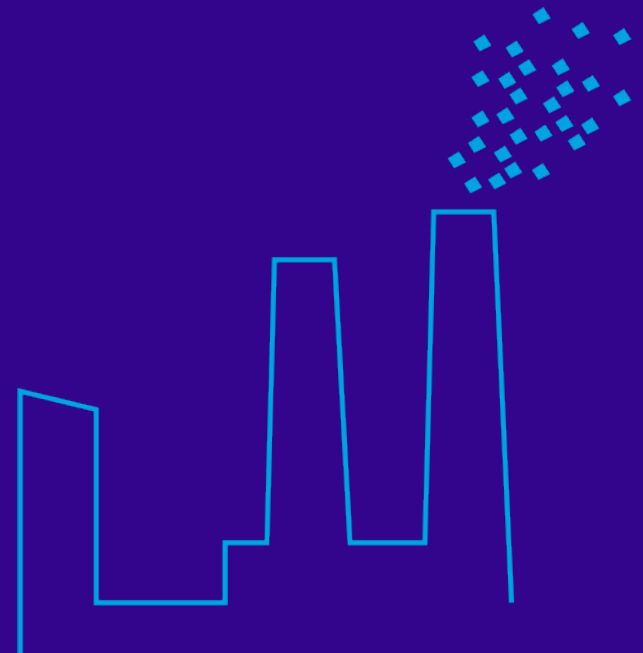
Campus Amiens

contact-amiens@lamanu.fr

https://lamanu.fr

70 rue des Jacobins, 80090 Amiens

09 86 27 17 04



LA MANU ÉCOLE SUPÉRIEURE DES MÉTIERS DU NUMÉRIQUE

NOVEI FORMATION

Campus Compiègne

contact-compiegne@lamanu.fr

https://lamanu.fr

Rue Robert Schuman – La Croix-Saint-Ouen I 60200 Compiègne
Entrée par le 41 rue Irène Joliot Curie (Bâtiment Millenium II)

09 86 27 17 04