Premiers programmes

1. La fonction print()

Nous avons rencontré dans les TP précédents la fonction **print()** qui affiche une chaîne de caractères. En fait, la fonction **print()** affiche l'argument qu'on lui passe entre parenthèses **et** un retour à ligne.

Une manière de s'en rendre compte est d'utiliser deux fonctions print() à la suite. Dans la portion de code suivante, le caractère « ; » sert à séparer plusieurs instructions Python sur une même ligne :

```
In [1]:  print("hello"),print("joe")
hello
joe
```

La fonction **print()** peut également afficher le contenu d'une variable quel que soit son type. Par exemple, pour un entier :

```
In [2]:  var=3

In [3]:  print(var)
3
```

Il est également possible d'afficher le contenu de plusieurs variables (quel que soit leur type) en les séparant par des virgules :

```
In [4]:  x=32

In [5]:  nom="John"

In [6]:  print(nom,"a",x,"ans")
John a 32 ans
```

Python a écrit une phrase complète en remplaçant les variables x et nom par leur contenu. Vous remarquerez que pour afficher plusieurs éléments de texte sur une seule ligne, nous avons utilisé le séparateur « , » entre les différents éléments.

Python a également ajouté un espace à chaque fois que l'on utilisait le séparateur « , ».

On peut modifier ce comportement en passant à la fonction print() l'argument par **mot-clé sep** :

```
In [8]:  print(nom,"a",x,"ans",sep="-")
John-a-32-ans
```

Pour afficher deux chaînes de caractères l'une à côté de l'autre, sans espace, on peut soit les concaténer (voir TP 2), soit par **mot-clé sep** avec une chaîne de caractères vide :

```
In [9]:  print(nom,"a",x,"ans",sep="")
Johna32ans
```

2. Premiers programmes

Point méthode

Vous devrez écrire sur papier vos programmes avant de le faire dans l'éditeur. .

(a) Inverser le contenu de deux variables

En vous référant au TP notions de variables, écrire dans un fichier inversion.py un script permettant d'échanger le contenu des variables a et b saisies par l'utilisateur.

Vous utiliserez les fonctions **`input()`** et **`print()`**.

Exemple Vous devrez obtenir dans le shell :

```
saisir a : 35
saisir b : prog
a=  prog
b=  35
```

(b) Calculer le cube d'un entier

Écrire un programme qui calcule et affiche le cube d'un entier saisi par l'utilisateur.

Exemple Vous devrez obtenir dans le shell :

```
saisir n : 8
le cube de 8 est égal à 512
```

```
saisir n : 6
le cube de 6 est égal à 216
```

(c) Conversion

Écrire dans l'éditeur un script qui demande à l'utilisateur d'entrer un temps t en secondes et qui le convertit en heures/minutes/secondes.

On affichera le résultat sous forme d'une phrase :

Exemple

```
saisir le temps en secondes : 567
567 secondes correspondent à 0 heures 9 minutes
et 27 secondes
```

(d) Un peu de mathématiques

Dans un repère, on donne deux points $A(x_A ; y_A)$ et $B(x_B ; y_B)$ tels que $x_A \neq x_B$.

Écrire dans l'éditeur un script qui renvoie l'équation (réduite*) la droite (AB).

* sous la forme $y = mx + p$

Exemple

```
abscisse du point A : 1
ordonnée du point A : 1
abscisse du point B : 5
ordonnée du point B : 5
la droite (AB) a pour équation réduite : y= 1.0 x + 0.0
```

3. Chaînes de caractères

(a) Longueur d'une chaîne

La fonction **len()** permet de calculer la longueur d'une chaîne de caractères c'est-à-dire son nombre de caractères :

```
In [40]:  ch="theodore-monod"

In [41]:  len(ch)
Out[41]: 14
```

(b) Accès aux caractères individuels d'une chaîne

Python offre une méthode simple pour accéder aux caractères contenus dans une chaîne : chaque caractère est accessible directement par son rang dans la chaîne en utilisant des crochets.

Attention comme souvent en informatique, le premier caractère de la chaîne est indexé par 0

Ainsi :

t	h	e	o	d	o	r	e	-	m	o	n	o	d
0	1	2	3	4	5	6	7	8	9	10	11	12	13

On obtient alors dans le shell :

```
In [37]:  ch="theodore-monod"

In [38]:  ch[4]
Out[38]: 'd'
```

Prévoir les sorties [35], [36] et [37]

- In [35]: ch[0]
- Out[35]:
- In [36]: ch[0] + ch[6] + ch[9]
- Out[36]:
- In [37]: print(ch[8]*2+ch[0])
- Out[37]:

Il est aussi possible d'indexer les caractères par des entiers négatifs ce qui revient à compter à partir de la fin (le dernier caractère de la chaîne est alors d'indice −1) :

t	h	e	o	d	o	r	e	-	m	o	n	o	d
-14	-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

Prévoir la sortie [38]

- In [38]: print(ch[−8] * 2 + ch[−5])
- Out[38]:

(c) Slicing

i. Principe

Plus généralement, la technique sur slicing (littéralement le découpage en tranches) permet d'extraire une portion d'une chaîne de caractères : il suffit de préciser l'**indice de début** i (qui sera **inclus** dans le découpage) et l'**indice de fin** j (qui sera **exclus**) sous la forme [i:j] .

Prévoir la sortie [40]

- In [40]: ch[3:7]
- Out[40]:

Si l'indice i n'est pas précisé, il est implicitement pris égal à 0 ; de même si j n'est pas précisé il sera égal à la longueur de la chaîne :

Prévoir la sortie [41]

- In [41]: ch[6:] + ch[:6]
- Out[41]:

ii. Extraire les k premiers/derniers caractères d'une chaîne

Il suffit d'écrire **ch[:k]**.

De même, pour extraire les k derniers caractères il suffit d'écrire **ch[−k:]** .

iii. Autres type d'extraction

Le slicing dans sa forme la plus générale prend un troisième paramètre correspondant au pas de la sélection :

ch[i:j:k] retourne tous les caractères dont les indices sont compris entre i (inclus) et j (exclus) et séparés d'un pas k.

Par exemple, pour obtenir les caractères d'indices pairs puis les caractères d'indices impairs, on écrira :

- In [42]: ch[::2]
- Out[42]:
- In [43]: ch[1::2]
- Out[43]:

iv. Inversion des caractères d'une chaîne

Le pas peut aussi prendre des valeurs négatives, ce qui nous donne un moyen simple d'inverser les caractères d'une chaîne :

In [44]: ch[::−1]

Out[44]:

(Lorsque le pas est négatif, les valeurs par défaut de i et j sont respectivement :

−1 et −len(ch)−1)

BILAN

Extraire une portion d'une chaîne de caractères ch	**ch[i:j]** **i inclus, j exclus**
Extraire les k premiers caractères	**ch[:k]**
Extraire les k derniers caractères	**ch[-k:]**
tous les caractères dont les indices sont compris entre i (inclus) et j (exclus) et séparés d'un pas k.	**ch[i:j:k]**
Inversion des caractères d'une chaîne	**ch[::−1]**

v. Application

Le mélange de Monge d'un paquet de cartes numérotées de 1 à 2n consiste à démarrer un nouveau paquet avec la carte 1, à placer la carte 2 au dessus de ce nouveau paquet, puis la carte 3 au dessous du nouveau paquet et ainsi de suite en plaçant les cartes paires au dessus du nouveau paquet et les cartes impaires au-dessous.

Autrement dit, si le paquet de cartes initial est représenté par la suite (1, 2, 3, . . . , 2n), son mélange de Monge sera représenté par la suite (2n, 2n-2, . . . , 4, 2, 1, 3, 5, . . . , 2n-2, 2n-1).

A. Déterminer le mélange de Monge de la suite (1, 2, 3, . . . , 8) ($n=4$). Bien détailler les étapes suivies.

B. Écrire en langage naturel une suite d'instructions permettant de réaliser ce mélange de Monge puis le programmer en Python.

C. Réaliser en une ligne Python un mélange de Monge d'un paquet de cartes numérotées de 1 à 8.