INTRODUCTION A LA PROGRAMMATION PYTHON

INTRODUCTION A LA PROGRAMMATION PYTHON
#1. Programme et Instructions
- Mission 1.1.
- Mission 1.2.
- Mission 1.3.
- Vidéo

#2. Variable et Type de donnée
- Notion de variables
  - Mission 2.1.
  - Mission 2.2.
- Type de données
  - Mission 2.3.
- Vidéo

> **Objectifs:**
>
> - Introduire Python pour les élèves de 2nd
> - Synthétiser les connaissances en programmation pour les élèves de 2nd et de 1ère spé Maths
> - Mettre en place des fondamentaux sur Python en début de 1ère spé NSI

> **Méthodologie:**
>
> - Lire l'activité
> - Réaliser les missions en programmant sur un environnement de développement:
>   - en ligne (présent notebook, repl.it, pythontutor , ...)
>   - installé sur votre ordinateur (idle de Python, Thonny, Pycharm Community, ...)
> - Visionner la vidéo de l'épisode pour plus d'explications et pour la correction

#1. Programme et Instructions

> **Objectifs:**
>
> - Comprendre ce qu'est un algorithme, un programme et une instruction
> - Construire un programme en Python avec des instructions de base

Un **algorithme** est une suite d'étapes à suivre pour atteindre un objectif. On peut l'écrire en langage naturel ou en pseudo code.

Un **programme** informatique est l'implémentation d'un algorithme dans un langage de programmation, il peut être exécuté dans une machine.

Un programme est constitué d'une suite finie d'**instructions**:

```
<instruction 1>
<instruction 2>
    . . .
<instruction n>  `
```

Ces instructions peuvent être de différentes natures, voici quelques exemples dans le langage Python:

```
################################
# Opération entre des données #
################################

2 + 2         # addition entre deux nombres entiers
2 * 2         # multiplication entre deux nombres entiers
2 ** 3        # puissance: 2 à la puissance 3

7 / 2         # division décimale entre deux entiers (le résultat n'est plus forcément un entier)
7 // 2        # quotient de la division euclidienne entre deux entiers
7 % 2         # reste de la division euclidienne entre deux entiers

'Python' + 'Introduction' # concaténation de deux chaînes de caractères (textes)
'Python' * 3              # concaténation de plusieurs fois la même chaîne de caractères


#################################################
# Appel à des services du système d'exploitation #
#################################################

print("Python Introduction")       # affichage d'un texte sur la sortie standard (l'écran)
print("2 x 2 = ", 2 * 2 )          # affichage d'un texte puis du résultat d'une expression arithmétique

input("Donnez votre réponse:")     # récupération d'un texte depuis l'entrée standard (le clavier)


#################################################
# Nous en verrons d'autres par la suite:        #
# - Affectation                                 #
# - Contrôle:                                   #
#       - condition                             #
#       - boucle bornée                         #
#       - boucle non bornée                     #
#################################################

```

Mission 1.1.

> Programmer une instruction qui affiche:
> $7 = 3 \times 2 + 1$
> en calculant les valeurs 3 et 1 avec des expressions arithmétiques

```
# Ecrire votre programme ici:


```

Mission 1.2.

> Programmer une instruction qui affiche un texte entré par l'utilisateur depuis le clavier

```
# Ecrire votre programme ici:


```

Mission 1.3.

> Programmer une instruction qui demande à l'utilisateur son nom, le récupère, puis l'affiche concaténé dix fois.

```
# Ecrire votre programme ici:


```

Vidéo

> https://youtu.be/zYhnN3jspUc

#2. Variable et Type de donnée

> **Objectifs:**
>
> - Introduire le concept de Variable
> - Programmer des instructions d'affectation
> - Différencier les types de données de base

Notion de variables

On a souvent besoin dans un programme de conserver une donnée en mémoire en vue d'un traitement utltérieur, on utilse pour cela une **variable** qui est représentée par un nom:

```
a = 3
```

Cette instruction est une instruction d'**affectation**: on affecte à la variable nommée ***a*** la valeur ***3***:

```
a = 3
print( a )
```

```
3
```

Pour faire simple, on peut voir une variable comme une boîte qui a un nom et qui contient une donnée.

Pour être plus précis, à une variable est associée une adresse mémoire et cette adresse mémoire contient la donnée.

On peut affecter à une variable directement une donnéec mais aussi le contenu d'une autre variable, le résultat d'une expresion arithmétique, la combinaison d'expressions, ... :

```
a = 3
a = a + 1    # on met dans a:  (l'ancienne valeur de a) + 1
b = a
b = b * 2    # on met dans b:  (l'ancienne valeur de b) * 2
c = a ** b
```

Mission 2.1.

> Déterminez la valeur des variables du programme ci-dessus:
>
> 1. sans exécuter le programme
> 2. en ajoutant des `print()` à la fin du programme et en l'exécutant

```
# Ecrire votre programme ici:
```

Mission 2.2.

> Ecrire un programme qui demande à l'utilisateur son nom, le récupère, puis lui demande son prénom, et enfin affiche un message de bienvenue contenant son nom et son prénom.

```
# Ecrire votre programme ici:
```

Type de données

Le **type** d'une donnée correspond à la nature de la donnée, nous en avons déjà vu certains en #1, voici la liste des types de base:

Type	Nature	Exemple	Opération
int (integer)	nombre entier relatif	2 1000000 -3 2020	+ (addition) * (multiplication) / (division décimale) ** (puissance) // (quotient division euclidienne) % (reste division euclidienne)
float (floating point)	nombre décimal relatif	2.3 -1.7245	+ (addition) * (multiplication) / (division décimale)
str (string)	chaîne de caractères (caractère, texte)	"Hello" "2020"	+ (concaténation) * (concaténation multiple)
bool (boolean)	booléen	True False	and (et logique) or (ou logique) not (non logique)

Pour compléter, on verra plus tard dans la série un dernier type: les listes.

Le type d'une variable est le type de la donnée affectée à la variable.

Il est très important de maîtriser la typologie car les opérations et d'autres manipulations ne sont pas les mêmes selon le type de la donnée:

```
# Initialisation des variables (première affectation)
nombre_a = 3
nombre_b = 5
string_a = 'texte a'
string_b = 'texte b'
booleen_a = True
booleen_b = False

# opérations correctes
nombre_a + nombre_b     # addition de deux nombres (résultat: 8)
```

```python
string_a + string_b     # concaténation de deux textes (résultat: 'texte atexte b')
booleen_a and booleen_b # et logique entre deux booléens (résultat: False)
booleen_a or booleen_b  # ou logique entre deux booléens (résultat: True)
not booleen_a           # non logique d'un booléen (résultat: False)

# opérations incorrectes: l'exécution entraîne une erreur (ou le résultat est inconsistant)
nombre_a + string_a     # + entre un nombre et une chaîne de caractère
nombre_a and nombre_b   # et logique entre deux nombres
not string_a            # non logique d'un texte
```

On peut connaître le type d'une variable avec la fonction: `type(nom_variable)`

```

On peut  changer le type d'une variable ( lorsque c'est possible) avec une opération de convertion (casting):


```python
# initialisation des variables
a = 3
print( type( a ) ) # l'instruction affichera int
b = '2.3'
print( type( b ) ) # l'instruction affichera str

# casting
c = str( a )       # la fonction str() convertit le nombre en string
print( type( c ) ) # l'instruction affichera str

d = int( c )       # la fonction int() convertit le texte en nombre entier (si c'est possible !)
print( type( d ) ) # l'instruction affichera int

e = float( b )     # la fonction float() convertit le texte en nombre décimal (si c'est possible !)
print( type( e ) ) # l'instruction affichera float

```

Mission 2.3.

> On reprend la mission 2.2. mais en plus:
>
> - on demande à l'utilisateur son année de naissance
> - on lui précisera son âge (en fin d'année) dans le message de bienvenue
>
> NB: la fonction `input()` renvoie toujours une string (et pas un nombre)
```

```
# Ecrire votre programme ici:
```

Vidéo

> https://youtu.be/krt_dZwXsJk