

PE1 Module 2

PE1 Module 2 - Python Data Types, Variables, Operators, and Basic I/O operations

M2 S1 - The "Hello, World!" Program

Code	Output
<code>print("Hello, world!")</code>	Hello, world!
<code>print("My name is", "Python")</code>	My name is Python
<code>print("Hello", "World", sep="*", end="!*")</code>	Hello,*World!*

M2 S2 - Python Literals

Data Type	Examples
Integer	-256, 256
Float	-256.2, 1.123e-10
String	"Hello", 'Hello World'
Boolean Values	True, False
Encodage	Output de 23 en décimal
décimal	<code>print(23)</code>
octal	<code>print(0o27)</code>
hexadécimal	<code>print(0x17)</code>
Exemple Escape character	Output
<code>print("C'est \"Jean-\n")</code>	C'est "Jean"
<code>print("He\'s "Jean")</code>	He's "Jean"

M2 S3 - Operators data manipulation tools

Operations	Exemples	Output
Addition	<code>print(2 + 2)</code>	4
Soustraction	<code>print(5 - 2)</code>	3
Multiplication	<code>print(2 * 3)</code>	6
Division	<code>print(6.3 / 3)</code>	2.1
Division entière	<code>print(6.3 // 3)</code>	2

M2 S3 - Operators data manipulation tools (cont)

Modulo	<code>print(9 % 6)</code>	3
Exposant	<code>print(2 ** 3)</code>	8

M2 S4 - Variables

Type of value to assign	Exemples
integer	<code>var = 1</code>
string	<code>name = "UwU"</code>
Expression	Shorcut operator
<code>i = i + 2 * j</code>	<code>i += 2 * j</code>
<code>var = var / 2</code>	<code>var /= 2</code>
<code>rem = rem % 10</code>	<code>rem %= 10</code>
<code>j = j - (i + var + rem)</code>	<code>j -= (i + var + rem)</code>
<code>x = x ** 2</code>	<code>x **= 2</code>

M2 S5 - Comments

Commentaires	<code># <texte></code>
--------------	------------------------------

M2 S6 - Interaction with the user

<code>input("Tell me anything")</code>	Inviter l'utilisateur à entrer quelque chose
<code>float(input("Tell me anything"))</code>	Caste l'entrée de l'utilisateur (qui est un string) en float

PE1: Module 3

PE1: Module 3 - Boolean Values, Conditional Execution, Loops, Lists and List Processing, Logical and Bitwise Operations

M3 S1 - Making decisions in Python

Comparaison	
Egalité	<code>==</code>
Inégalité	<code>!=</code>
Plus grand que	<code>></code>
Plus petit que	<code><</code>
Plus grand ou égal à	<code>>=</code>

M3 S1 - Making decisions in Python (cont)

Plus petit ou égal à	<code><=</code>
Condition d'exécution	
Si	<code>if <condition>:</code>
Sinon si	<code>elif <condition>:</code>
Sinon	<code>else <condition>:</code>

M3 S2 - Looping your code with while

Boucles	
Tant que	<code>while <condition>:</code>
Infini tant que	<code>while True:</code>
Pour i de 0 à 100	<code>for i in range(100):</code>
Pour i de 2 à 7 en incrémentant par 3	<code>for i in range(2, 8, 3):</code>

Instructions dans la boucle

Sortir de la boucle	<code>break</code>
Passer directement à la prochaine itération de la boucle	<code>continue</code>

M3 S3 - Logic and bit operations in Python

Operateurs logiques	
Et	<code>and</code>
Ou	<code>or</code>
negation	<code>not</code>
exemple	<code>not (p and q) == (not p) or (not q)</code>
Operateurs binaires	A peut être 0 ou 1 B peut être 0 ou 1
Disjonction (or) binaire	<code> </code> <code>A B</code>

M3 S3 - Logic and bit operations in Python (cont)

Conjonction (and) binaire &
A & B

Ou exclusif (xor) binaire ^
A ^ B

Negation (not) binaire ~
~A

Operateurs de décalage (shift)
Pour var = 17

Décaler vers la droite revient à diviser par deux

Décaler vers la gauche revient à multiplier par deux

```
var_right = var >> 1      print(var_
                           _right)
                           8
```

```
var_left = var << 2      print(var_left)
                           68
```

M3 S4 - Lists

Initialisation et indexation

Initialisation d'une liste
numbers = [10, 5, 2, 1]
print(numbers)
Output : [10, 5, 2, 1]

Modification d'un élément
numbers[1] = 3
print(numbers)
Output : [10, 3, 2, 1]

Accès à un élément négativement
print(numbers[-2])
Output : 2

Différences

Fonctions result = fonction(arg)
Méthodes result = data.method(arg)

M3 S4 - Lists (cont)

Fonctions

Fonction len
Calculer la taille de la liste
print(len(numbers))
Output : 4

Fonction del
Supprimer un élément de la liste.
del numbers[0]
print(numbers)
Output : [3, 2, 1]

Méthodes

Méthode append
Ajouter un élément à la fin de la liste
numbers.append(4)
print(numbers)
Output : [3, 2, 1, 4]

Méthode insert
Ajouter un élément sur un emplacement défini
numbers.insert(0,5)
print(numbers)
Output : [5,3,2,1,4]

M3 S5 -Sorting simple lists

Trier une liste avec la méthode sort

Initialisation de la liste
my_list = [8, 10, 6, 2, 4]

Affichage de la liste
print(my_list)
Output : [8, 10, 6, 2, 4]

Application de la fonction
my_list.sort()

Affichage de la liste
print(my_list)
Output : [2, 4, 6, 8, 10]

Il faut voir le **tri à bulle**

M3 S6 - Operations on lists

Slices :
Afficher à une partie de la liste
my_list[start:end]

M3 S6 - Operations on lists (cont)

Afficher du 1er élément au 3ème
print(my_list[:3])
Output : [10,8,6]

Afficher du 3ème élément au dernier
print(my_list[2:])
Output : [6,4,2]

Afficher du 2ème élément au 3ème
print(my_list[1:3])
Output : [8,6]

Supprimer des éléments 2 à 3
del my_list[1:3]
print(my_list)
Output : [10,4,2]

Opérateurs in et not

in
Vérifier si un élément est présent dans la liste
elem in my_list

not in
Vérifier si un élément n'est pas présent dans la liste
elem not in my_list

On utilise la liste [10, 8, 6, 4, 2]

M3 S7 - Lists in advanced applications

Créer une ligne d'échec de pion blanc en une ligne de code

```
row = [WHITE_PAWN for i in range(8)]
```

Créer un plateau d'échec avec une liste à deux dimensions en une ligne de code

```
board = [[EMPTY for i in range(8)] for j in range(8)]
```

