Cheatography

Python PCEP Cheat Sheet

by Freebeater via cheatography.com/177378/cs/37029/

PE1 Module 2

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

M2 S1 - The "Hello, World!" Program

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

M2 S2 - Python Literals

DataType	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	print(23)
octal	print(0o27)
héxadécimal	print(0x17)
Exemple Escape character	**Output**
print("C'est \"Jean-\"")	C'est "Jean"
print('He\'s "Jean"')	He's "Jean"

M2 S3 - Operators data manipulation tools

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

M2 S3 - Operators data manipulation tools (cont)

Modulo	print(9 % 6)	3
Exposant	print(2 ** 3)	8

M2 S4 - Variables

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

M2 S5 - Comments

Commentaires	# <texte>

M2 S6 - Interaction with the user

input("Tell me anything")	Inviter l'utilisateur à entrer quelque chose
float(input("Tell me anything"))	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	==
Inégalité	!=
Plus grand que	>
Plus petit que	<
Plus grand ou égal à	>=

M3 S1 - Making decisions in Python (cont)

Plus petit ou égal à	<=
Condition d'exécution	
Si	`if <co ndi tio - n>:`
Sinon si	`elif <co ndi - tio n>:`
Sinon	`else <co ndi - tio n>:`

M3 S2 - Looping your code with while

Boucles	
Tant que	`while <co ndi - tio n>:`
Infini tant que	`while True:`
Pour i de 0 à 100	`for i in range( - 100):`
Pour i de 2 à 7 en incrémentant par 3	`for i in range(2, 8, 3):` range(start, stop, step)
Instructions dans la boucle	
Sortir de la boucle	break
Passer directement à la prochaine itération de la boucle	continue

M3 S3 - Logic and bit operations in Python

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

Python PCEP Cheat Sheet

by Freebeater via cheatography.com/177378/cs/37029/

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**	Output
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 = function(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

Methode **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 3eme	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)]