Pensée Computationnelle et programmation

I. L'algorithme, les structures de données et les structures simples

1. L'algorithme :

1.1- Qu'est-ce qu'un algorithme ?

Un algorithme est …….
permettant de résoudre un problème.

1.2- Squelette d'un algorithme :

Pour écrire un algorithme il faut respecter la syntaxe suivante :

Algorithme nom

Début

Traitement

Fin

Déclaration des objets :

Objet	Rôle

1.3- Qu'est-ce qu'un Langage de programmation ?

Un **langage de programmation** est un ensemble de règles et de syntaxes qui permet aux programmeurs de donner des instructions à un ordinateur pour effectuer des tâches spécifiques. En d'autres termes, c'est un moyen de communiquer avec un ordinateur pour lui indiquer quoi faire.

Actictivité1 : écrire l'algorithme d'un programme qui permet de calculer la surface d'un cercle.

les données (input)→ Algorithme→ résultat(output)

Les données sont les entrées de l'algorithme utilisées pour produire la solution du problème à résoudre.

2-Les structures de données :

2.1- Les constantes :

Une constante est un objet ……

.2.2- Les variables :

2.2.1- Qu'est-ce qu'une variable ?

En programmation, pour résoudre un problème informatique, il faut utiliser des variables. Chaque variable possède un ……., un …….. et une …….

Les valeurs des variables peuvent être de plusieurs types.

2.2.2- Les types de données

2.2.2.1Types numériques

a) Le type entier(int):

- Une variable de type entier contient une valeur sans virgule : 2, -17, 1254, 0 , -5

Les opérateurs applicables sur le type entier :

Opération	En algorithmique	En Python
Somme	+	+
Soustraction	-	-
Multiplication	*	*
Division	/	/
Division entière	**Div**	//
Reste de la division entière	**Mod**	%

Opération	En algorithmique	En Python
Egal	=	==
Différent	≠	!=
Strictement supérieur	>	>
Supérieur ou égal	≥	>=
Strictement inférieur	<	<
Inférieur ou égal	≤	<=
Appartient (**entier**, **caractère**)	∈	**in**

b/le type réel (float)

Une variable de type réel contient une valeur avec virgule : 5.2 , -7.0 , 2.55, 15.

Les opérateurs applicables sur le type réel :

Opération	En algorithmique	En Python
Somme	+	+
Soustraction	-	-
Multiplication	*	*
Division	/	/

Opération	En algorithmique	En Python
Egal	=	==
Différent	≠	!=
Strictement supérieur	>	>
Supérieur ou égal	≥	>=
Strictement inférieur	<	<
Inférieur ou égal	≤	<=

Les fonctions prédéfinies sur les types numériques :

Syntaxe Algorithmique	Syntaxe Python	Rôle de la fonction	Exemples
Abs(x)	**abs(x)**	Donne la valeur absolue de x	Abs(-12.5) = 12.5
Ent(x)	**int(x)**	Supprime la partie décimale	Ent(15.63) = 15 Ent(-4.5) = -4
Arrondi(x)	**round(x)**	Donne l'entier le plus proche **NB :** si la partie décimale =0.5 la fonction retourne l'entier pair le plus proche	Arrondi(12.9) = 13 Arrondi(-4.2)=-4 Arrondi(1.5) = 2 Arrondi(2.5)=2
Racine_carrée(x)	**sqrt(x)**	Retourne la racine carrée d'un nombre positif	Racine_carré(16)= 4.0
Aléa(vi,vf)	**randint(vi,vf)**	Donne un entier aléatoire de l'intervalle [vi, vf]	Aléa(10,202) = 15 Aléa(0,1) = 1

- En python pour utiliser la fonction sqrt : `from random import randint`
- En python pour utiliser la fonction randint : `from random import randint`

2.2.2.2Le type booléen :

- Une variable logique ou booléenne (bool) ne peut contenir que la valeur Vrai (True) ou

Faux (False)

- La table de vérité du type booléen est la suivante :

Opération	En algorithmique	En Python
Négation	**Non**	**not**
Conjonction	**Et**	**and**
Disjonction	**Ou**	**or**

x	y	non(x)	x et y	x ou y
Vrai	**Vrai**	**Faux**	**Vrai**	**Vrai**
Vrai	**Faux**	**Faux**	**Faux**	**Vrai**
Faux	**Vrai**	**Vrai**	**Faux**	**Vrai**
Faux	**Faux**	**Vrai**	**Faux**	**Faux**

2.2.2.3 Le type caractère :

- Une variable de type caractère doit contenir un seul caractère.
- Un caractère peut être : "a", "b", …,"A", "B", … ,"0", … ,"9", "*", "=", …
- Chaque caractère possède un code ASCII.

Exemples :

HEX	DEC	CAR	HEX	DEC	CAR	HEX	DEC	CAR
20	32		40	64	@	60	96	`
21	33	!	41	65	A	61	97	a
22	34	"	42	66	B	62	98	b
23	35	#	43	67	C	63	99	c
24	36	$	44	68	D	64	100	d
25	37	%	45	69	E	65	101	e
26	38	&	46	70	F	66	102	f
27	39	'	47	71	G	67	103	g
28	40	(	48	72	H	68	104	h
29	41	)	49	73	I	69	405	i
2A	42	*	4A	74	J	6A	106	j
2B	43	+	4B	75	K	6B	107	k
2C	44	,	4C	76	L	6C	108	l
2D	45	-	4D	77	M	6D	109	m
2E	46	.	4^E	78	N	6^E	110	n
2F	47	/	4F	79	O	6F	111	o
30	48	0	50	80	P	70	112	p
31	49	1	51	81	Q	71	113	q
32	50	2	52	82	R	72	114	r
33	51	3	53	83	S	73	115	s
34	52	4	54	84	T	74	116	t
35	53	5	55	85	U	75	117	u
36	54	6	56	86	V	76	118	v
37	55	7	57	87	W	77	119	w
38	56	8	58	88	X	78	120	x
39	57	9	59	89	Y	79	121	y
3A	58	:	5A	90	Z	7A	122	z
3B	59	;	5B	91	[	7B	123	{
3C	60	<	5C	92	\	7C	124	|
3D	61	=	5D	93	]	7D	125	}
3E	62	>	5E	94	^	7E	126	~
3F	63	?	5F	95	_	7F	127	

Les fonctions prédéfinies sur le type caractère

En algorithmique	En Python
Ord (c)	**ord** (c)
Chr (d)	**chr** (d)

Exemples
ord("B") = 66 ord("@") = 64 ord("1") = 49
chr(48) = "0" chr(65)= "A"

2.2.2.4 Le type chaine de caractères :

- Une variable de type chaine de caractères peut contenir un ensemble de caractères
- Une chaine vide est une chaine ayant une longueur égale à zéro
- L'indice du premier caractère d'une chaine est zéro.

Les fonctions prédéfinies sur le type chaine de caractères :

Syntaxe Algorithmique	Syntaxe Python	Rôle de la fonction	Exemples
long(ch)	**len(ch)**	Donne le nombre de caractères de la chaine ch	long("prog") = 4
pos(ch1,ch2)	**ch2.find(ch1)**	Donne la 1ère position de ch1 dans ch2.	pos("o","foot") = 1
convch(d)	**str(d)**	Convertit un nombre décimal en chaine	convch(123)="123"
valeur(ch)	**int(ch)** **float(ch)**	Convertit une chaine en valeur numérique	valeur("12.9") = 12.9
estnum(ch)	**ch.isdigit()**	Donne Vrai si la chaine est convertible en une valeur numérique, sinon elle donne Faux	estnum("2598") = Vrai estnum("25 98") = Faux .
sous_chaine (ch,d,f)	**ch[d :f]**	Retourne une partie de la chaine ch à partir de la position d jusqu'à la position f-1	sous_chaine("Python", 0,4)="Pyth"
efface(ch,d,f)	**ch[:d]+ch[f :]**	Retourne la chaine formée par la partie de la position d jusqu'à la position f-1	efface('Python', 1,4) ="Pon"
majus(ch)	**ch.upper()**	Retourne la chaine en majuscule	majus("Python") = "PYTHON"

3. Les structures simples :

Appelées aussi les opérations algorithmiques simples (ou de base) et qui sont :

➢ L'opération ……

➢ L'opération ……

➢ L'opération ……..

L'opération d'entrée	
Notation Algorithmique	**Notation en Python**
Ecrire (" Message ") **Lire** (Objet)	*Pour les chaînes de caractères :* Objet = **input** ("Message ") *Pour les entiers :* Objet = **int** (**input** ("Message ")) *Pour les réels :* Objet = **float** (**input** ("Message "))

L'opération de sortie	
Notation Algorithmique	**Notation en Python**
Ecrire ("Message", Objet) **Ecrire** ("Message", Expression)	**print** ("Message", Objet, **end** = "") **print** ("Message", Expression, **end** = "")
Ecrire_nl ("Message", Objet) **Ecrire_nl** ("Message", Expression)	**print** ("Message", Objet) **print** ("Message", Expression) ***N.B. :*** *"print" fait un retour à la ligne automatique*

→La commande Ecrire : affiche sans faire un retour à la ligne

→La commande Ecrire_nl : affiche et fait un retour à la ligne

II. Les structures de contrôles conditionnelles

Une structure conditionnelle permet d'exécuter certaines instructions uniquement si une condition est remplie.

1. Structure conditionnelle simple

1.1 Structure conditionnelle simple réduite

En algorithmique	En python
Si Condition **Alors** Traitement **FinSi**	**if** Condition **:** Traitement

1.2 Structure conditionnelle simple alternative (complète)

En algorithmique	En python
Si Condition **Alors** Traitement1 **Sinon** Traitement2 **FinSi**	**if** Condition **:** Traitement1 **else :** Traitement2

2. Structure conditionnelle généralisée

En algorithmique	En python
Si Condition1 **Alors** Traitement1 **Sinon Si** Condition2 **Alors** Traitement2 …… [**Sinon** TraitementN] **FinSi**	**if** Condition1 **:** Traitement1 **elif** Condition2 **:** Traitement2 …… **else :** TraitementN

3. Structure conditionnelle à choix

Activité1 :

Ecrire un algorithme qui permet de saisir un entier entre 0 et 10 et de l'afficher en toutes lettres.

Exemple : 5 Cinq

En utilisant la structure conditionnelle généraliséee	En utilisant la structure conditionnelle à choix

Activité2 : Ecrire un algorithme qui à partir un numéro de mois, permet d'afficher la saison.

Exemple : 5 printemps

En utilisant la structure conditionnelle généraliséee	**En utilisant la structure conditionnelle à choix**

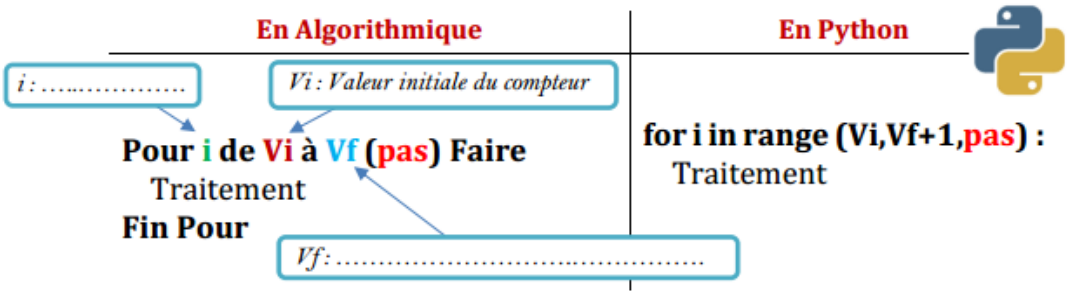
En algorithmique	En python
Selon <Sélecteur> Valeur1_1[, Valeur1_2, …] : Traitement1 Valeur2_1 . . Valeur2_2 : Traitement2 …… **[Sinon** TraitementN] **Fin Selon**	**A partir de la version 3.10** **match** Sélecteur : **case** Valeur1 **:** Traitement1 **case** Valeur2_1 | Valeur2_2 **:** Traitement2 **case** Sélecteur if V3_1 <=Sélecteur<= V3_2 **:** Traitement3 **case** _ **:** TraitementN ***N.B.* : Le sélecteur doit être de type scalaire.**

En Python (contrairement aux autres langages de programmation) c'est l'indentation (Les espaces en début de chaque ligne) qui détermine les blocs d'instructions (structures conditionnelles, boucles, etc.)

III. Les structures de contrôles itératives

Les structures itératives ou structures répétitives ou ……. permettent de ……. Un traitement un certain nombre de fois. En algorithmique on peut distinguer 3 boucles :

1- La boucle « Pour » : la structure itérative complète permet de répéter un ensemble d'instructions un nombre fini et connu à l'avance d'instructions.



Remarque:

- **Lorsque le compteur ne vérifie pas la condition d'entrée à la boucle, la boucle ne sera pas exécutée.**
- La fonction python « range » crée un compteur qui s'incrémente ou se décrémente automatiquement.

Exemple :

- range(n) renvoi un itérateur parcourant 0, 1, 2 ... , n − 1 ;
- range(n,m) renvoi un itérateur parcourant n, n+1, n+2, ..., m − 1 ;
- range(n,m,p) renvoi un itérateur parcourant n, n+p, n+2p , ..., m − 1.

➔ **P ici est le pas**
➔ Le « Pas » peut être Positif ou négatif, par défaut, le « Pas » est égal à 1
➔ Le parcours peut être croissant (Début > Fin) ou décroissant (Fin<Début) dans les deux cas, il faut faire attention au « Pas ».