Les différentes méthodes d'implémentation en python

1) *Les bibliothèques utilisées*

```
from math import*
from random import*
from numpy import*
from pickle import*
```

2) *Les opérateurs*

En algorithme	En python
+, -, *, /, div, mod	+, -, *, /, //, %
non, et, ou	not, and, or (*ou bien* : and : & ; or : |)
>, ≥, <, ≤, =, ≠	>, >=, <, <=, ==, !=
∈ **(entier ou caractère)** Exemple : si x ∈ ['A','Z'] alors si x ∈ ['A'. .'Z'] alors si n ∈ [10 . . 99] alors	**in** Exemple : if x in ['A','Z'] : if x in range(ord('A'),ord('Z')+1) : **ou** if 'A'<= x<='Z' : if n in range(10,100) : **ou** if 10<= n<=99 :

3) *Les fonctions prédéfinies*

En algorithme	En python
Arrondi(x)	round(x)→**NB :** round(x.**50**)donne l'entier paire le plus proche de x
RacineCarré(x)	sqrt(x)
Ent(x)	int(x)
Abs(x)	Abs(x)
Aléa(VALi, VALf)	randint(VALi, VALf)
Chr(x)	chr(x)
Ord(c)	ord(c)
Long(ch)	len(ch)
Pos(ch1, ch2)	ch2.find(ch1)
Convch(x)	str(x)
Estnum(ch)	ch.isdecimal() **ou** ch.isdigit() **ou** ch.isnumeric()
Valeur(ch)	int(ch) *pour la conversion en un entier* float(ch) *pour la conversion en un réel*
Sous_chaine(ch, d, f)	ch[d : f]
Effacer(ch, d, f)	ch[: d] + ch[f :]
Majus(ch)	Ch.upper()

4) Les structures de données (enregistrements, tableaux et matrices)

En algorithme	En python
Eleve=enregistrement mat : chaine [6] np : chaine [30] moy : réel fin	Eleve={'mat' :str,'np' :str,'moy' :float}
T=tableau de 30 entiers T=tableau de 30 chaines T=tableau de 30 eleves	T=array([int]*30) T=array([str]*30) T=array([eleve]*30) **ou** T=array([{}]*30)
M=tableau de 10*20 des réels	M=array([[float]*20]*10)

5) Les structures simples

En algorithme	En python
Lire(x) *Lire un champ d'enregistrement :* Lire(x.champ)	x= input() x['champ']= input()
Ecrire_nl(x) *afficher un champ d'enregistrement :* ecrire_nl(x.champ)	Print(x) Print(x['champ'])
Ecrire(x)	Print(x, end='sep')
x←valeur *affecter un champ d'enregistrement :* x.champ← valeur	x=valeur x['champ']=valeur

6) La structure selon (avec python 3.10)

En algorithme	En python
Selon x Val1 : Instruction(s) 1 Val2, val3 **:** Instruction(s) 2 Val4 . . val5 : Instruction(s) 3 …….. **Autres :** Instruction(s) n **FinSelon**	**match** x : case Val1 : Instruction(s) 1 case Val2 | val3 : Instruction(s) 2 case x if val4<= x<= val5 **:** Instruction(s) 3 case _ : Instruction(s) n

7) *Les structures itératives*

En algorithme	En python
Pour : compteur de type caractère Pour i de 'A' à 'Z' faire ***Pour : les indices du tableau de type caractère*** Pour i de 'A' à 'Z' faire T[i]←…….	For i in range (ord('A'),ord('Z')+1) : For i in range(26) : T[i]=……
TantQue (condition) **Faire** Instruction(s) **FinTantQue**	**while** condition **:** Instruction(s)
Répéter Instruction(s) **Jusqu'à** (condition)	Initialisation : (si on a besoin :exemple dans le contrôle de saisie) **while not** (condition) **:** Instruction(s)

8) *Les fichiers textes*

En algorithme	En python
Ouvrir("chemin\Nom_physique", Nom_Logique, "mode ") Mode : "r " : Lecture "w" : Ecriture (création) "a" : Ajout	Nom_Logique=open ("chemin\Nom_physique", "mode ") Mode : "r " : Lecture "w" : Ecriture (création) "a" : Ajout
Lire(Nom_Logique, ch)	Nom_Logique.read()
Lire_ligne(Nom_Logique, ch)	Nom_Logique.readline()
Ecrire(Nom_Logique, ch)	Nom_Logique.write(ch)
Ecrire_nl(Nom_Logique, ch)	Nom_Logique.write(ch+"\n")
Fermer(Nom_Logique)	Nom_Logique.close()
Fin_fichier(Nom_Logique)	**Parcourir un fichier texte :** Ch=F.readline() While ch !='': ch=ch[:-1] traitement ch=F.readline()

9) *Les fichiers binaires*

En algorithme	En python
Ouvrir ("chemin\Nom_physique", Nom_Logique, "mode ") Mode : "rb " : Lecture "wb" : Ecriture (création) "ab" : Ajout	Nom_Logique=open("chemin\Nom_physique", "mode ") Mode : "rb " : Lecture "wb" : Ecriture (création) "ab" : Ajout
Lire(Nom_Logique, Var)	Var= load(Nom_Logique)
Ecrire(Nom_Logique, Var)	dump(Var, Nom_Logique)
Fermer(Nom_Logique)	Nom_Logique.close()
Fin_fichier(Nom_Logique) 	**Parcourir un fichier binaire :** Fin=False While Fin==False: try: traitement except: Fin=True

10) Les modules

Les fonctions

Algorithme	Python
Fonction Nom_fonction (pf$_1$: typ$_1$; pf$_2$: typ$_2$; …) **:** type_fonction **Début** Instruction(s) **Retourner** Var_result **Fin**	**def** Nom_fonction (pf$_1$, pf$_2$, …) **:** Instruction(s) **return** Var_result

Les procédures

Algorithme	Python
Procédure Nom_proc (pf$_1$: typ$_1$; pf$_2$: typ$_2$; … ; @ pf$_3$: typ$_3$; @ pf$_4$: typ$_4$; …) **Début** Instruction(s) **Fin**	**def** Nom_proc (pf$_1$, pf$_2$, …, pf$_3$, pf$_4$, …) **:** Instruction(s) **return** pf$_3$, pf$_4$ { si pf$_3$, pf$_4$ de types simples (objets non mutables}

Important

Python *: Tout objet de type composé (tableau, matrice, enregistrement, fichier) hérite automatiquement les modifications subies dans le corps du module appelé.*