2024

Programmation

Sofianne Nabli
ITITCH

Avant-propos

Sommaire

La forme générale d'un algorithme:

ALGORITHME Nom

DEBUT

  Traitements

FIN

Tableau de déclaration des objets (T.D.O)	
Objet	Type/Nature

Les syntaxes des structures algorithmiques:

Les opérations élémentaires simples

- L'opération d'entrée

Algo	Python
Ecrire(‘commentaire’) Lire(objet)	Pour les chaines de caracteres: Objet = input(‘Commentaire’) Pour les entiers: Objet = int(input(‘Commentaire’)) Pour les reels: Objet = float(input(‘Commentaire’))

- L'opération de sortie

Algo	Python
Écrire ("Message", Objet) Écrire ("Message", Expression)	print(‘ Message ’, Objet, end=’’) print(‘ Message ’, Expression , end=’’)
Écrire_nl ("Message", Objet, Expression)	print(‘ Message ’, Objet) print(‘ Message ’, Expression)

- L'opération d'affectation

Algo	Python
Objet ← valeur Objet ← Expression Objet1 ← objet2	Objet = valeur Objet = Expression Objet1 = objet2

- Les types de données simples

Type de donnée en algo	Type de donnée en python
Entier	int()
Réel	float()
Booléen	bool()
Caractère	str()
Chaîne de caractères	str()

Exemples de conversion entre les types simples en python

Conversion	Syntaxe	Exemple
De str vers int	int(ch)	x = int('3') x recoit 3
De str vers float	float(ch)	x = float('3.2') x recoit 3.2
De str vers bool	bool(ch)	x = bool('0') x recoit True
De int vers str	Str(int)	X = str(3) x recoit '3'

Ex1 : a = 10 b = 15 c = 30 a = -b b = a * 3 + b * 2 c = b + c / a • Quelle est la valeur de a et b et c après • exécution des actions suivantes	Réponse : A = -15 B = -15 C = -17
Ex2: On souhaite écrire un algorithme qui permet de calculer et afficher le montant à payer par un client qui a acheté n claviers sachant que le prix d'un clavier est 14 Dinars et qu'il a une remise de 15 %. Compléter l'algorithme ci-dessous par les instructions suivantes :	rem <== (mont * 15) / 100 Ecrire ("le montant à payer est : ", mont) mont <== mont - rem Lire (n) mont <== 14 * n

Les declarations:

Les objets de type de données simples:

Tableau de déclaration des objets (T.D.O)	
Objet	Type/Nature
Nom_constante	Constante = valeur de la constante
Nom_variable	Type_variable

NB:

L'indice du 1er caractere d'une chaine de caractere est zero

Pour acceder a un caractere d'une chaine ch, on utilise la notation ch[i] avec 0 <= i <= long(ch) - 1

On pourra utiliser l'operateur + pour concatener deux chaines.

Designation	Priorité	Notation		Type operande
Parentheses	1	( )	( )	Tous les types
Multiplication	2	*	*	Entier ou réel
Division réelle		/	/	Réel
Division entiére		Div	//	entier
Reste de la division		Mod	%	entier
Addition	3	+	+	Entier ou réel
Soustraction		-	-	Entier ou réel
Egale	4	=	==	Tout type ordonné
Différent		≠	!=	Tout type ordonné
Strict superieur		>	>	Tout type ordonné
Superieur ou egal		≥	>=	Tout type ordonné
Strict inferieur		<	<	Tout type ordonné
Inferieur ou égal		≤	<=	Tout type ordonné
Appartenance(entier, caractere ou booléen)		∈	in	Type scalaire

Appartenance	Algo	Python
Ensemble	x ∈ [val1, val2, ..., valn]	x in [val1, val2, ..., valn]
Intervalle	x ∈ [val1.. valn] Ou val1 ≤ x ≤ valn	Pour les entiers: x in range(val1, valn+1) val1 <= x <= valn Pour les caracteres ord(x) in range(ord(val1), ord(valn) + 1) val1 <= x <= valn

Opération	Algo	Python	priorite	Table de verité		
Négation	NON	not	1	A	Not(A)	
				True	False	
				False	True	
Conjonction	ET	and	2	A	B	A and b
				True	True	True
				True	False	False
				False	True	False
				False	False	False
Disjonction	OU	or	3	A	B	A or B
				True	True	True
				True	False	True
				False	True	True
				False	False	False

Les fonctions sur les caractéres:

Algo	Python	Role	Exemple	Résultat
ord(caractere)	ord(caractere)	Retourne le code ASCII du caractere	ord('A') ord('a')	65 97
chr(x)	chr(x)	Retourne le caractére dont le code ASCII x	chr(65) chr(97)	'A' 'a'

Les fonctions prédéfinies:

Algo	Python	Role	Exemple	résultat
abs(x)	abs(x)	Retourne la valeur absolue	abs(-20) abs(-5.8)	20 5.8
valeur(x)	int(x)	Retourne la partie entiere	int(5.2) int(-5.8)	5 -5
arrondi(x)	round(x)	Retourne l'entier le plus proche de x. En python si la partie fractionnaire est egale a 5, l'entier pair le plus proche est retourné	round(2.2) round(2.8) round(2.5) round(3.5)	2 3 2 4
racinecarré(x)	From math import sqrt N = sqrt(x)	Retourne la racine carré de x Si x < 0 elle provoque une erreur	sqrt(9) sqrt(25.0) sqrt(-5)	3.0 5.0 erreur
aléa(vi, vf)	From random import randint N = randint(vi, vf)	Retourne un entier aleatoire entre vi et vf	randint(2,5)	2 ou 3 ou 4 ou 5

Les structure de controle condionnelles:

Algo	Python
si conditon alors Traitement fin_si	if conditon : Traitement
si conditon alors Traitement 1 Sinon Traitement 2 fin_si	if conditon : Traitement else: Traitement2
si conditon1 alors Traitement 1 Sinon si conditon2 alors Traitement 2 ..... Sinon Traitement N fin_si	if conditon : Traitement elif conditon2 : Traitement2 ..... else: Traitement2

Algo	Python
Selon <Selecteur> Valeur1: Traitement1 Valeur2: Traitement2 Sinon TraitementN FinSelon	match selecteur: case valeur1: Traitement1 case valeur2 : Traitement2 case valeur3 : Traitement3 case _: TraitementN

Exemple

On se propose d'afficher la nature d'un caractère donné (consonne, voyelle, chiffre, symbole). Écrire un algorithme "Nature" correspondant à la résolution de cette situation puis l'implémenter en Python.

```
ALGORITHME Nature
DEBUT
  Ecrire ("Saisir un caractère : ")
  Lire (c)
  c ← Majus (c)
  Selon c
    "A" .. "Z" : Selon c
                   "A", "E", "I", "O", "U", "Y" : Ecrire ("Voyelle")
                   Sinon
                      Ecrire ("Consonne")
                 FinSelon
    "0" .. "9" : Ecrire ("Chiffre")
    Sinon
       Ecrire ("Symbole")
  FinSelon
FIN
```

T.D.O	
Objet	**Type/Nature**
c	Caractère

```
c=input("Saisir un caractère : ")
c=c.upper()
match c :
   case c if "A"<= c <="Z" :
      match c :
         case "O"|"I"|"Y"|"E"|"A"|"U":
              print("Voyelle")
         case _ :
              print("Consonne")
   case c if "0" <= c <= "9":
         print("Chiffre")
   case _ :
         print("Symbole")
```

Les structures de controle itératives:

En algo
pour compteur **de** début **a** fin [pas = valeur_pas] **faire** Traitement **fin pour**

En python
for compteur **in range**(debut, fin + 1, pas) : traitement

NB: la valeur finale est exclue de la boucle

- La valeur de pas peut etre positive ou negative, par defaut elle est egale a 1
- Ne pas utiliser l'instruction break pour forcer l'arret de la boucle for

La structure de contrôle itérative à condition d'arrêt (Répéter ... Jusqu'à ...)

Algo	Python
répéter Traitement **jusqu'à** Condition(s) de sortie	Pas de correspondance. Toutefois, on peut utiliser : valide = False while not valide: Traitement valide = (Condition(s) de sortie)

La structure de contrôle itérative à condition d'arrêt (Tant que ... Faire)

Algo	Python
tantQue Condition **Faire** Traitements **finTantQue**	while Condition: Traitements

NB:

- En Python, il est conseillé d'éviter l'utilisation de l'instruction « break », « continue » et « pass » dans les structures conditionnelles et les structures itératives.

Les modules:

a. La déclaration:

<table>
<tr><th>Algo</th><th>Python</th></tr>
<tr><td>Fonction Nom_fonction (pf1 : type1,
pf2 : type2, …,
pfn : typen) : Type_résultat
DEBUT
Traitement
Retourner résultat
FIN</td><td rowspan="2">Un module (fonction ou procédure) se définit en utilisant le mot-clé def selon la syntaxe suivante :

def Nom_module(pf1, pf2, …, pfn):
Traitement
[return résultat]

N.B. : Dans un module, l'instruction return peut être utilisée, et ce, pour retourner un seul résultat de type simple.</td></tr>
<tr><td>Procédure Nom_procedure (pf1 : type1,
pf2 : type2, …,
pfn : typen)
DEBUT
Traitement
FIN</td></tr>
</table>

b. L'appel:

Type	Algo	Python
Fonction	Objet ← Nom_fonction(pe1, …, pen)	Objet = Nom_module(pe1, …, pen)
Procédure	Nom_procedure(pe1, …, pen)	Nom_module(pe1, …, pen)

c. Le mode de passage:

En algorithmique : Si le mode de passage est par référence (par adresse), on ajoutera le symbole @ avant le nom du paramètre.

Algo	Python
Procédure Nom_procedure (@pf1 : type1, @pf2 : type2, …, pfn : typen) DEBUT Traitement FIN	Nom_module(pf1, pf2, …, pfn): Traitement **N.B.** : En Python, les paramètres de type dictionnaire, tableau et fichier sont, par défaut, passés par référence.

d. La portée des variables en Python :

Toute variable déclarée au sein d'un module a une portée locale.

Toute variable déclarée au sein d'un module précédée par le mot-clé global a une portée globale. Par conséquent, elle ne devra pas figurer parmi les paramètres de ce module.

Si nous avons plusieurs paramètres de même type et qui ont un passage par adresse, ils doivent être précédés par « @ ».

Par exemple : PROCÉDURE Traitement (@ A, B : Entier, X, Y : Réel).

En Python, pour résoudre le problème de passage par adresse, on peut suivre l'une des deux démarches suivantes :

La 1ère démarche :

Ne pas mettre les paramètres formels passés par adresse dans l'entrée de la procédure.

Mettre les paramètres formels passés par adresse dans le corps de la procédure précédés du mot-clé global.

La 2ème démarche :

Ne pas mettre les paramètres formels passés par adresse dans l'entrée de la procédure.

Utiliser le mot return pour retourner les valeurs des paramètres formels passés par adresse (au niveau algorithmique).

L'exemple ci-après illustre le passage par adresse en algorithmique et en Python.

<table>
<tr><td colspan="2">Notation en algo</td></tr>
<tr><td>Déclaration de la procédure saisir</td><td>L'appel de la procédure saisir</td></tr>
<tr><td>Procédure Saisir (@ n : entier)
Début
  Répéter
  Écrire ("Saisir un entier entre 5 et 20 : ")
  Lire (n)
  Jusqu'à (5 ≤ n ≤ 20)
Fin</td><td>Saisir(n)</td></tr>
<tr><td colspan="2">Notation en Python de la 1ère démarche</td></tr>
<tr><td>Déclaration de la procédure saisir</td><td>L'appel de la procédure saisir</td></tr>
<tr><td>def Saisir():
  global n
  valid = False
  while valid == False:
  n = int(input("Saisir un entier entre 5 et 20 : "))
  valid = (5 <= n <= 20)</td><td>Saisir()</td></tr>
<tr><td colspan="2">Notation en Python de la 2ème démarche</td></tr>
<tr><td>Déclaration de la procédure saisir</td><td>L'appel de la procédure saisir</td></tr>
<tr><td>def Saisir():
  valid = False
  while valid == False:
  n = int(input("Saisir un entier entre 5 et 20 : "))
  valid = (5 <= n <= 20)
  return n</td><td>n = Saisir()</td></tr>
</table>

Les tableaux:

En algo: Tableau a une dimension

Objet	Type/nature
Nom_tableau	Tableau de N Type_élément

On utilisera la bibliothèque numpy pour implémenter les tableaux.

Un tableau de la bibliothèque numpy est :

- Homogène, c'est-à-dire constitué d'éléments de même type.
- Statique, car sa taille est fixée lors de la création.

La déclaration d'un tableau se fait en deux étapes :

Importation des modules nécessaires de la bibliothèque numpy

```
from numpy import array
ou
from numpy import *
ou
import numpy as alias
```

Déclaration du tableau

```
T = array([Type_élément] * N)
ou bien
T = array([valeur_initiale] * N)
```

Remarque :

On peut spécifier le type des éléments d'un tableau avec la syntaxe :

Nom_tableau = array([Valeur_initiale] * N, dtype=Type_élément)

Exemples de déclarations de tableaux en Python:

Declaration	Explication
T = array([5] * 10)	Déclarer un tableau T de 10 entiers et initialiser ses éléments par « 5 ».
T = array([float()] * 10)	Déclarer un tableau T de 10 réels et initialiser ses éléments par « 0.0 ».
T = array([str] * 10)	Déclarer un tableau T de 10 chaînes de caractères.
T = array([str()] * 10)	Déclarer un tableau T de 10 caractères et initialiser ses éléments par le caractère vide.

Exemple d'un programme en Python présentant une solution modulaire

```
from numpy import array
T1 = array([0]*15) # Déclaration du tableau T1
T2 = array([0]*15) # Déclaration du tableau T2


# Définition du module saisieTaille
def saisieTaille(bornInf, bornSup):
    taille = 0
    while taille not in range(bornInf, bornSup+1):
        taille = int(input("Taille entre " + str(bornInf) + " et " + str(bornSup) + " : "))
    return taille


# Définition du module remplirTab
def remplirTab(T, taille):
    for i in range(taille):
        T[i] = int(input("Donner l'élément N° " + str(i) + " : "))
```

```
# Définition du module afficherTab
def afficherTab(T, taille):
    for i in range(taille):
        print(T[i])


# Le programme principal
n = saisieTaille(5, 10) # 1er appel du module saisieTaille
m = saisieTaille(3, 15) # 2ème appel du module saisieTaille
print("Chargement de T1")
remplirTab(T1, n)
print("Chargement de T2")
remplirTab(T2, m)
print("Affichage du tableau T1")
afficherTab(T1, n)
print("Affichage du tableau T2")
afficherTab(T2, m)
```

2ème façon d'implémentation du module saisieTaille en utilisant une variable globale nommée Taille

```
from numpy import array
T1 = array([0]*15) # Déclaration du tableau T1
T2 = array([0]*15) # Déclaration du tableau T2
# Définition du module saisieTaille
def saisieTaille(bornInf, bornSup):
    global taille
    taille = 0
    while taille not in range(bornInf, bornSup+1):
        taille = int(input("Taille entre " + str(bornInf) + " et " + str(bornSup) + " : "))
# Définition du module remplirTab
def remplirTab(T, taille):
    for i in range(taille):
        T[i] = int(input("Donner l'élément N° " + str(i) + " : "))

# Définition du module afficherTab
def afficherTab(T, taille):
    for i in range(taille):
        print(T[i])
```

```
# Le programme principal
saisieTaille(5, 10) # 1er appel du module saisieTaille
n = taille
saisieTaille(3, 15) # 2ème appel du module saisieTaille
m = taille
print("Chargement de T1")
remplirTab(T1, n)
print("Chargement de T2")
remplirTab(T2, m)
print("Affichage du tableau T1")
afficherTab(T1, n)
print("Affichage du tableau T2")
afficherTab(T2, m)
```

Exercice: Remplir puis afficher un tableau t par n entiers donnés avec 5 <= n <= 10

Algo

```
Procédure Saisir (@ m : entier)
Début
    Répéter
        Écrire ("Taille du tableau : ")
        Lire (m)
    Jusqu'à ∈[5..10]
Fin
Procédure Remplir (@ v : tab, m : entier)
Début
    Pour i de 0 a m - 1 faire
        Écrire ("v[" , i , "] = ")
        Lire (v[i])
    FinPour
Fin
```

TDOL	
Objet	Type/nature
i	entier

```
Procédure Afficher (v : tab, m : entier)
Début
    Pour i de 0 a m - 1 faire
        Écrire (v[i])
    FinPour
Fin
```

TDOL	
Objet	Type/nature
i	entier

Python

```
from numpy import array

def Saisir():
    valide = False
    while valide == False:
        m = int(input("Taille du tableau : "))
        valide = m in range(5, 11)
    return m

def Remplir(m):
    v = array([int(0)] * m)
    for i in range(m):
        v[i] = int(input("T[" + str(i) + "] = "))
    return v

def Afficher(v, m):
    for i in range(m):
        print(v[i], end=" ")

# Programme principal
n = Saisir()
t = Remplir(n)
Afficher(t, n)
```

#Algorithme du programme principal

ALGORITHME Exemple_Tableau

Début

Saisir (n)

Remplir (t,n)

Afficher (t,n)

Fin

TDNT
Type
Tab = tableau de 10 entiers

TDO	
Objet	**Type/nature**
m	entier
t	tab
Saisir	procédure
Remplir	procédure
Afficher	procédure

Fonctions sur les chaines de caracteres:

En algorithmique	En Python	Rôle	Exemple
Lo←**long** (Ch)	Lo = **len** (Ch)	Retourne un entier représentant le nombre de caractères de la chaîne **Ch** (la longueur de Ch).	Lo = len ("Salut") → Lo == 5 Lo = len ("L'élève") → Lo == 7 Lo = len ("") → Lo == 0
Po←pos (Ch1, Ch2)	Po = Ch2.**find** (Ch1)	Retourne un entier représentant la position de la 1ère occurrence de **Ch1** dans **Ch2**. Elle retourne **-1** si **Ch1** n'existe pas dans **Ch2**.	Ch1 = "Y" Ch2 = "BAYBAY" Po = Ch2.find (Ch1) → Po == 2
Ch2 ←**sous_chaîne** (Ch1, Début, Fin)	Ch2 = Ch1 **[**Début : Fin**]**	Retourne une copie de la chaîne **Ch1** à partir de l'indice **Début** à l'indice **Fin** (position **Fin** exclu).	Ch1 = "BACCALAUREAT" Ch2 = Ch1 [5 : 12] **Chr == "LAUREAT"**
Ch2 ←**effacer** (Ch1, d, f)	Ch2= Ch1**[** : d**]**+Ch1 **[**f: **]**	Retourne une chaîne **Ch2** après avoir effacer, de la chaîne **Ch1**, les caractères de la position **d** à la position **f** (**f** exclu).	Ch1 = "INFORMATIQUE" Ch2 = Ch1[:6] + Ch1[11:] **Ch2 == "INFORME"**
ChM←**majus** (Ch)	ChM = Ch.**upper** ( )	Retourne la chaîne **ChM** représentant la conversion en Majuscule de la chaîne **Ch**.	Ch = "Jour" ChM = Ch.upper ( ) → ChM =="JOUR
Ch←**convch** (**X**)	Ch = **str** (**X**)	Retourne la conversion du nombre **X** en une chaîne de caractères.	N = 358 Ch = str (N) → Ch == "358"
Test←**estnum** (Ch)	**Pas de correspondance**. Toutefois, on pourra utiliser **isnumeric** ( ) malgré qu'elle ne répond pas aux exigences ou bien développer un module qui permet de réaliser cette tâche.	Retourne **VRAI** si la chaîne **Ch** est convertible en une valeur numérique et **FAUX** dans le cas contraire.	Ch = "489" Test = Ch . isnumeric ( ) Test == True Test == False
N←**valeur** (Ch)	N = **int** (Ch) *ou bien* N = **float** (Ch)	Retourne la conversion d'une chaîne **Ch** en une valeur numérique, si c'est possible.	Ch = "489" N = int (Ch) → N == 489 Ch = "489" N = float (Ch) → N == 489.0

Exercices:

Exercice 1:

Ecrire un programme qui permet de déterminer et d'afficher le nombre de chiffres d'un entier donné.

Exemples :

N = 49 le programme affichera : Nombre de chiffres = 2

N = 2013 le programme affichera : Nombre de chiffres = 4

Exercice 2:

Ecrire un programme qui permet de saisir deux entiers m et n.

· Concaténer l'entier m avec l'entier n

· Affecter le résultat de concaténation a une variable p puis afficher le résultat de concaténation.

Exemple : m = 167, n =25 **le programme affichera :** p =16725

Exercice 3 :

Écrire une fonction somme qui prend en argument une chaîne de caractères comprenant des entiers séparés par des symboles +, comme « 7+52 » et qui renvoie le résultat de la somme.

a)on suppose que le format de la chaine est correcte on a entier1+entier2

b) on suppose que le format de la chaine est correcte on a objet1+objet2

Exercice 4 :

Ecrire un programme qui permet de saisir une valeur horaire H de type chaine sous la forme suivante :

"hh-mm" de modifier le format de l'heure en "hh:mm" puis de la convertir en une seule valeur en secondes.

Exemples :

Donner l'heure :"01-10" le programme affichera :

Nouveau format : "01 :10"

Le nombre de secondes est : 4200

Exercice 5 :

Ecrire un programme qui permet de saisir une date de la forme jj/mm/année

Puis d'afficher le message suivant :

Jour :jj

Mois :mm

Année :année

Exemple d'execution

Donner la date

22/12/2020

Votre date est :

Jours :22

Mois :12

Année :2020

Solutions:

Exercice1:

```
n = int( input("donner n :") )
ch = str(n)
print("le nombre de chiffres est :", len(ch))
```

Exercice2:

```
n = int( input("donner n :") )
m = int( input("donner m :") )
ch = str(n) + str(m)
print("le nouveau entier est :", int(ch))
```

Exercice 3

```
algorithme ex3
début
    ecrire("donner la formule : ")
    lire(exp)
    op <--  pos("+", ch)
    nb1 <-- sous_chaîne(exp, 0, op)
    nb2 <-- sous_chaîne(exp, op+1, long(ch))
    somme <-- valeur(nb1) + valeur(nb2)
    ecrire(somme)
Fin
```

TDO	
Objet	**Type/nature**
exp,nb1,nb2	chaine
somme	entier

Python

1)

```
exp = input("donner la formule : ")
op = exp.find("+")
nb1 = exp[0:op]
nb2= exp[op+1:]
somme = int(nb1) + int(nb2)
print(somme)
```

2)

```
valid = False
while valid == False:
    exp = input("donner la formule : ")
    op = exp.find("+")
    nb1 = exp[0:op]
    nb2= exp[op+1:]
    if nb1.isnumeric() and nb2.isnumeric():
        valid = True
somme = int(nb1) + int(nb2)
print(somme)
```

Exercice 4

Algo:

```
algorithme ex4
début
    ecrire("donner la heure hh-mm ")
    lire(h)
    trait <--  pos("-", h)
    heures <-- sous_chaîne(h, 0, trait)
    minutes <-- sous_chaîne(h, trait+1, long(h))
    ch <-- heures + ":" + minutes
    ecrire("le nouveau format est ", ch)
    secondes <-- valeur(heures) * 60 * 60 + valeur(minutes) * 60
    ecrire("le nombre en secondes est ",secondes)
fin
```

Python

```
h = input("donner l'heure hh-mm : ")
trait = h.find("-")
heures = h[0:trait]
minutes = h[trait+1:]
ch = heures + ":" + minutes
print("le nouveau format est ",ch)
secondes = int(heures) * 60 * 60 + int(minutes) * 60
print("le nombre en secondes est ",secondes)
```

Exercice 5

Algo	Python
algorithme ex5 début ecrire("donner la date jj/mm/annee : ") lire(datte) jour <-- sous_chaîne(datte, 0, 2) mois <-- sous_chaîne(datte, 3, 5) annee <-- sous_chaîne(datte, 6, long(datte)) ecrire("votre date est : \njour :", jour) ecrire("mois :", mois) ecrire("annee :", annee) fin	datte = input("donner la date jj/mm/annee : ") jour = datte[0:2] mois = datte[3:5] annee = datte[6:] print("votre date est : \njour :", jour) print("mois :", mois) print("annee :", annee)

Exercice N°1 :

Donner les déclarations en algorithmique puis en Python des constantes suivantes :

pi = 3.14, g=9.8, e=2.718

Objet	Type/nature

En python

……
……
……
….

Exercice N°2 :

a) Remplir le tableau suivant :

Description Algorithmique	Traitement Algo	Type Algo	Traitement Python	Type Python
Affecter la valeur 15 à la variable a				
Affecter la valeur 25 à la variable C1				
Affecter la valeur "5" à la variable Num				
Affecter la valeur "3Info2" à la variable classe				
Affecter la valeur Vrai à la variable test				

b) Comment appelle-t-on cette instruction ?

.....

Exercice N°3 :

Soient les séquences algorithmiques suivantes, donner la valeur de chaque objet :

N°	Séquence	a	b	c
1	a ← 2 b ← a			
2	a← 3 b← 8 b ← a a ← b			
3	a← 5.2 b ← 2.1 c ← a a← b b← c			
4	a← 6 b← a+5			

N°	Séquence	a	b	c
	c← b+1			
5	a←34 mod10			
	b← 34div 10			
	a← 123456789 mod 10			
	b← 123456789 div 10			
6	a← 5			
	b← 2			
	c←a/ b			
7	a← 8			
	b← 3			
	c← a=b			

Exercice N°4 :

Soient les variables x et y de type entier dont les valeurs sont les suivantes : x=2,y=4 Compléter le tableau suivant :

Expression	Résultat	Type du résultat
a ← x + y + 1		
b ← "x" + "y" + "1"		
c ← pos ("x", "y mod x")		
d ← arrondi (2.54) + ord (chr (122))		
e ← valeur ("112" + "33")		
f ← convch (Aléa (0,1))		
g ← "a" + "a" = "2a"		
h ← long ("Python") mod 2 = 0		
i ← pos ("Jour", "Bonjour")		
j ← convch (ord ("A") + 3)		
k ← estnum (j)		
l ← "10b" + convch (76)+"?h"		
m ← valeur(l)		
n ← chr (99) < majus ("m")		

Exercice N°5 : Compléter le tableau suivant :

Expression	Résultat	Type du résultat
a ← len ("Sciences de l'Informatique") % 11 // 3		
b ← not (155 < 99) or (ord ("D") == 1) and (chr (97) == "c")		
c ← "a"		
ch ← ch.upper () < chr (ord ("d"))		
d ← "F" <= "B" and (round (1.85) > 0)		
ch ← "20"		
e ← ("A" > "C") or ch.isdigit ()		
f ← str (32 % 4 + int ("658")) // 4		
ch ← "1245" g ← int (str (209) + "1") + ch.find ("4")		

Exercices Les structures conditionnelles

Exercice 1 :
Écrire un algorithme qui permet de lire un chiffre (compris entre 0 et 9) puis afficher ce nombre en toutes lettres.
Exemple
entrée = 4, sortie = quatre

Exercice 2 :
Soient x et y deux variables entières, écrivez un algorithme qui vérifie si x est divisible par y ou non ; les deux variables sont lues au clavier.

Exercice 3 :
Écrire un algorithme qui lit un caractère et affiche si oui ou non c'est une lettre de l'alphabet.

Exercice 4 :
Écrire un algorithme qui lit un entier et affiche un message pour dire s'il est positif, négatif ou nul.

Exercice 5 :
Écrire un algorithme qui lit les paramètres d'une équation de premier degré ax + b = 0 et affiche la solution.

Exercice 6 :
Écrire un algorithme permettant de lire la valeur de la température de l'eau et d'afficher son état :

- "Glace" si la température est inférieure à 0, t < 0
- "Eau" si la température est strictement supérieure à 0 et inférieure à 100, 0 < t < 100
- "Vapeur" si la température est strictement supérieure à 100, t > 100

Exercice 7 :
Le taux d'intérêt bancaire pour un montant déposé à la banque dépend du temps pendant lequel le montant a été déposé.
Voici un tableau présentant le taux selon le nombre d'années de dépôt.

Années en dépôt	**Taux d'intérêt**
Durée > 5 ans	0,095
5 >= durée > 3	0,085
3 >= durée > 1	0,065
1 >= durée	0,058

Écrivez un algorithme qui lit le nombre d'années de dépôt et affiche le taux d'intérêt.

Exercice 8 :
Écrivez un algorithme qui permet de saisir un numéro de couleur correspondante :

1. Rouge
2. Orange
3. Jaune
4. Vert
5. Bleu
6. Indigo
7. Violet

Exercice 9 :
Écrire un algorithme d'un programme qui permet de vérifier la parité d'un entier n donné. (pair ou impair)

Exercice 10 :
Écrire un algorithme d'un programme qui permet de saisir un entier n et vérifier s'il est multiple de 5.

Exercice 11 :
Un nombre n est dit cubique s'il est égal à la somme des cubes de ses chiffres.
Exemple : n=153=13+53+33 n = 153 = 1^3 + 5^3 + 3^3 n=153=13+53+33
Écrire un algorithme qui permet de saisir un entier n, en supposant qu'il est formé de trois chiffres, et indique s'il est cubique ou non.

Exercice 12 :
Écrire un algorithme qui permet de saisir un entier n, on suppose qu'il est formé de quatre chiffres, et indique s'il est symétrique ou non.
Exemple :

- Si n=7575 , ce nombre n'est pas symétrique
- Si n=4114, ce nombre est symétrique

Solution les structures conditionnelles

Exercice 1

Algorithme exercice 1

Début

Répéter (écrire "donner un chiffre compris entre 0 et 9")

Lire(x)

Jusqu'à (x>=0 et x<=9)

Selon (x) faire

0 : écrire ("zéro")

1 : écrire("un")

2 : écrire("deux")

3 : écrire("trois")

4 : écrire("quatre")

5 : écrire("cinq")

6 : écrire("six")

7 : écrire("sept")

8 : écrire("huit")

9 : écrire("neuf")

Fin

TDO

Objet	Type/nature
x	entier

Exercice 2

Algorithme exercice2

Début

Ecrire ("donner deux entiers")

Lire(x,y)

Si(y=0) alors

Ecrire ("impossible de diviser par 0")

Sinon si (x mod y=0) alors

Ecrire (y, "divise"x)

Sinon

Ecrire ("y,ne divise pas"x)

Finsi

Fin

TDO

Objet	Type/nature
X,y	entier

Exercice 3:

Algorithme : ex3

Début

Écrire ("saisir un caractère")

Lire(c)

Si (c >= 'A' et c <= 'Z') alors

Écrire (c, " est majuscule")

Sinon

Si (c >= 'a' et c <= 'z') alors

Écrire (c, " est minuscule")

Sinon écrire (c, "n'est pas une lettre alphabet")

finsi

finsi

Fin

objet	type/nature
c	caractère

Exercice 4:

Algorithme : signe

Début

Écrire ("donner un entier")

Lire(n)

Si (n > 0) alors

Écrire (n, "est positif")

Sinon si (n < 0) alors

Écrire (n, "est négatif")

Sinon

écrire (n, "est égal à 0")

finsi

finsi

Fin

objet	type/nature
n	entier

Exercice 5:

Algorithme : ex5

Début

Écrire ("donner deux entiers")

Lire (a, b)

Si (a = 0) alors

Écrire("impossible")

Sinon

Écrire ("la solution est", -b/a)

finsi

finsi

Fin

objet	type/nature
a, b	entier

Exercice 6

Début

Écrire("donner la température")
lire(temp)

si (temp ≤ 0) alors
  Écrire("Glace")
sinon si (temp > 0 et temp ≤ 100) alors
  Écrire("Eau")
sinon
  Écrire("Vapeur")
fin si
fin

objet	type/nature
temp	réel

Exercice 7

Algorithme : ex7

Début
Écrire("donner la durée")
lire(durée)

objet	type/nature
durée	entier

si (durée > 5) alors
  Écrire("0.095")
sinon si (3 < durée ≤ 5) alors
  Écrire("0.085")
sinon si (1 < durée ≤ 3) alors
  Écrire("0.065")
sinon si (durée ≤ 1) alors
  Écrire("0.058")
fin si
fin

Exercice 8

Algorithme : ex8

Début
répéter
  Écrire("donner un numéro entre 1 et 7")
  lire(num)
jusqu'à (num ≥ 1 et num ≤ 7)

objet	type/nature
num	entier

Selon (num) faire

1. écrire("rouge")
2. écrire("orange")
3. écrire("jaune")
4. écrire("vert")
5. écrire("bleu")
6. écrire("indigo")
7. écrire("violet")

Fin selon

Fin

Exercice 9:

Algorithme : parite

Début

Écrire ("donner un nombre")

Lire(n)

objet	type/nature
n	entier

Si (n mod 2 = 0) alors

Écrire (n, " est pair")

Sinon

Écrire (n, " est impair")

Fin si

Fin

Exercice 10:

Algorithme : ex10

Début

Écrire ("donner un entier")

Lire(n)

objet	type/nature
n	entier

Si (n mod 5 = 0) alors

Écrire ("le nombre est multiple 5")

Sinon

Écrire ("le nombre est non multiple 5")

Fin si

Fin

Exercice 11:

Algorithme : ex11

Début

Écrire ("donner un entier")

Lire(n)

a <--- n div 100

b <--- (n mod 100) div 10

c <--- n mod 10

objet	type/nature
n, a, b, c	entier

si (n = a^3 + b^3 + c^3) alors

Écrire ("le nombre est cubique")

Sinon

Écrire ("le nombre n'est pas cubique")

Fin si

Fin

Exercice 12

Algorithme exercice 12

Début

Ecrire ("donner un nombre")

Lire(n)

A←----n div1000

b←---(n mod 1000) div 100

c←---n mod 100 div 10

d←---n mod 10

objet	Type/nature
N,a,b,c,d	entier

Si(a=d) et (b=c) alors

Ecrire ("le nombre est symétrique")

Sinon

Ecrire ("le nombre est non symétrique")

Finsi

Fin

Exercice N°6

Écrire un algorithme d'un programme permettant de saisir le rayon R d'un cercle puis calculer et afficher sa surface.

Sachant que la surface d'un cercle = πR^2

Exercice N°7

Écrire un algorithme qui permet de saisir une chaîne de caractères en représentant le nom et prénom d'un utilisateur puis générer un mot de passe pour les utilisateurs d'une application informatique. Le mot de passe est généré comme suit :

La longueur de la chaîne formée par le nom et le prénom suivi par le caractère qui suit le premier caractère du nom et le dernier caractère dans l'ordre alphabétique suivi par le code ASCII du dernier caractère du nom et prénom.

Mot de passe généré :

Mot de passe = Longueur + Caractere suivant le premier + Code ASCII du dernier caractere

Mot de passe="10N110"

Exercice N°8

Écrire un programme python qui simule le jet de deux dés à six faces, on additionne le résultat des deux dés puis on l'affiche sur l'écran.

Exercice N°9

Pour traduire un mot français en latin, on place la première lettre du mot français à la fin et on ajoute « us ».
Exemple :

Le mot **homme** → devient **ommehus**, en outre omme + us.

Écrire l'algorithme d'un programme qui permet de réaliser cette traduction.

Exercice N°10

Écrire un programme python permettant de saisir une durée de communication D de la forme « mm:ss » puis calculer le frais de communication sachant que le second sera facturé à 5 millimes.

Exemple :
D = « 02:26 » alors le programme affiche « Frais de communication : 730 millimes »

En effet : 730 = (2*60+26)*5

Exercices: Les structures de contrôle itératives

Exercice N°1 : Évaluer les expressions suivantes

A)

x←8 Répéter x←x+2 y←x*2 jusqu'à y>25	p←0 tant que p <5 faire p←p+2 fin tant que	p←0 tant que p >5 faire p←p+5 fin tant que	b←1 pour i de 2 a 6 faire b←b*i fin pour
X= Y=	P=	P=	B=
Nombre d'itérations =	Nombre d'itérations =	Nombre d'itérations =	Nombre d'itérations =

B)

x←val répéter x←x+2 k←x*2 Jusqu'à k>30	X←Val répéter x←x+2 k←x*2 Jusqu'à k<30	X←5 Y←8 Z←3 Pour i de 1 à 4 faire Si Y < 10 alors Y ← Y + 2 Sinon X ←X + 2 Fin Si Fin Pour	X←8 Y←10 Z←5 Pour i de 8 a 5 (pas=-1)faire y←y+i Fin pour Z←Z+1
val←5 X= Y=	val←5 X= Y=	X= Y= Z=	X= Y= Z=
Nombre d'itérations =	Nombre d'itérations =	Nombre d'itérations =	Nombre d'itérations =

Exercice N°2 :

Soit la séquence algorithmique suivante

Algorithme traitement

```
Début :
    Ecrire ("x ="), Lire (x)
    Ecrire ("y ="), Lire (y)
  P ← 1
  Pour i de 1 à y faire
     P ← P * x
  Fin Pour
 Ecrire(P)
Fin
```

Questions :

1. Faire le traçage à la main de cet algorithme avec x = 5 et y = 3.

i=				
p=				

2. Quel est le rôle de cette séquence algorithmique ?

Exercice 3:

Soit la séquence suivante :

Répéter

Ecrire (“x=“)

Lire (x)

Jusqu’à (x mod 2 = 1) et (x > 10)

Quelle est le rôle de la séquence d’instruction suivante ?

.....

Quelle est la valeur saisie de x permettant d’arrêter la répétition ?

X=	22	9	8	35	16	19	23
Arrêt							

Exercice 4:

Soit la sequence algorithmique suivante:

Algorithme traitement 1

Debut

Ecrire (‘CH=’)

Lire (CH)

Répeter

P←POS(‘/’,CH)

CH← efface(ch,p,p+1)

Jusqu’a (Pos(’/’,CH)=-1)

Ecrire (CH)

Fin

Questions

1) Faire le tournage a la main de cet algorithme avec CH’01/10/2022’

P=					
CH=					

2) Quel est le role de cet sequence algorithmique

.....

3) Remplacer la structure repeter...jusqu’a par la structure tant que...faire

.....

Exercice N°5 :
Écrire un algorithme qui calcule et affiche la somme suivante (avec **n** un entier impair).
S=1+1/2+1/3+1/4+…+1/n

Exercice N°6 :
Écrire un algorithme permettant de saisir une chaîne de caractères **CH commençant par une lettre majuscule et se terminant par un point** puis de calculer et d'afficher le nombre de mots dans cette chaîne.

NB : Deux mots consécutifs peuvent être séparés par un ou plusieurs espaces.

Exercice 7 :
Écrire un programme permettant de saisir n (20 > n > 5) puis remplir un tableau T par n chaînes de caractères de longueur >= 3 puis calculer et afficher la somme S comme suit :

N=6

T=

AB33P?	Dre7lp	Mo;zt	145	RT22S	SD!!

S=33+7+0+145+22+0=207

Solution ex7:

```
from numpy import *


b = False

while not b:

    n = int(input("donner n : "))

    b = n > 5 and n < 20


t = array( [str] * n )


for i in range( n ):

    c = False

    while not c:

        t[i] = input("donner chaine : ")

        c = len(t[i]) >= 3
```

```
s= 0

for i in range( len(t) ):

    num=""

    char = t[i]

    for j in range(len(char)):

        if char[j].isnumeric():

            num += char[j]

    if len(num) != 0:

        s += int(num)


print(s)
```

Sous-programmes

Exercice:

Soit l'algorithme de la F1 suivant :

Fonction F1(X : .....):.....

Début

K ← ""

Pour i de 0 à N - 1 faire

Si X[i] ∈ { "0", "9" } alors

K ← K + X[i]

Fin Si

Fin Pour

.....

Fin

a. Complétez les pointillés ainsi que le tableau de déclaration des objets locaux de F1 :

Objet	Type

b. Quelle est la valeur de la variable Z après chacune des affectations suivantes ?

Affectation	Valeur de Z
Z ← F1("Bac2023")	
Z ← F1("2TechnologieS3")	

c. Déduire le rôle de la fonction F1 :

.....

.....

Exercice:

Soit les algorithmes ci-dessous correspondant à un programme principal Exercice et à une fonction Inconnue appelée par celui-ci :

Algorithme Exercice	Fonction Inconnue(C : .....) : .....
Début Lire(A) Si (Inconnue(A) = A) alors Écrire(A, "Vérifie la propriété.") Sinon Écrire(A, "Ne vérifie pas la propriété.") Fin Si Fin	Début S ← 0 Pour i de 1 à C div 2 faire Si (C mod i = 0) alors S ← S + i Fin Si Fin Pour Retourner S Fin

a. À partir des algorithmes ci-dessus, remplir la 2e colonne du tableau suivant par un exemple de chaque cas cité dans la 1re colonne :

Elements	Exemples
Instruction d'initialisation	
Paramètre formel	
Paramètre effectif	

b. Compléter l'entête de la fonction Inconnue par les types appropriés :

Fonction Inconnue(C : .....) : .....

c. Completer le tableau des objets locaux de la fonction inconnue

Objet	type

d. Parmi les variables A, C, S, i quelles sont celles qui ne sont pas visibles par le programme principale:

.....

e. En deduire le role de la fonction inconnue:

.....

NB: pour chaque exercice une solution modulaire est exigée

Exercice:

On se propose d'écrire un algorithme permettant de générer automatiquement des mots de passe pour les utilisateurs d'une application informatique, en suivant les étapes suivantes :

On remplit un tableau par les noms de N utilisateurs (avec $2 \leq n \leq 9$), sachant qu'un nom d'utilisateur est formé de 20 lettres majuscules au maximum :

On génère un tableau TM contenant les mots de passe des N utilisateurs. Un mot de passe est généré en apportant les modifications suivantes au nom de l'utilisateur :

- Remplacer tous les occurences du caractére A par le caractère @.
- Remplacer tous les occurences du caractére O par le caractère 0.
- Ajouter à la fin de la chaîne obtenue le nombre de voyelles contenus dans le nom d'utilisateur.

Exemple:

Pour le tableau des utilisateurs suivant :

RAOUF	ZERIEB	AZIZA	FATMA	RAYEN	NADIA

Le tableau de mot de passe sera:

R@0UF3	Z@RIEB3	@ZIZ@3	F@TM@2	R@YEN3	N@DI@3

Exercice:

On se propose d'écrire un algorithme permettant de sécuriser l'envoi des messages entre deux chercheurs en utilisant une clé de cryptage selon le principe suivant :

a. Saisir le message à crypter msg, sachant qu'il est composé par des lettres majuscules et des espaces.

b. Saisir la clé de cryptage qui est une chaîne de caractères composée uniquement par des caractères numériques et ayant la même longueur que msg.

c. Remplacer chaque lettre du message d'ordre alphabetique i par la lettre d'ordre alphabetique j avec j=i+c sachant que c est le chiffre de la chaine chcle ayant le meme indice que la lettre à crypter.

NB : L'espace ne sera pas crypté. Si d+cl e>26, on reprend les lettres dès le début.

Exemple:

Soit le message "EXCELLENTE PERFORMANCE" et soit la clé "195462378401653628451"

Msg	E	X	C	E	L	L	E	N	T	E		P	E	R	F	O	R	M	A	N	C	E
chcle	1	9	5	4	6	3	2	7	3	8	4	0	1	6	5	3	6	2	8	4	5	1
Msgc	F	G	H	I	R	O	G	U	W	M		P	F	X	K	R	X	O	I	R	H	F

Regles:

La lettre E est d'ordre alphabetique 5, elle sera remplacée par la lettre d'ordre alphabetique 5+1=6, soit F.

La lettre X est d'ordre alphabetique 24, elle sera remplacée par la lettre d'ordre alphabetique 24+9=33. ==> 33 mod 26=7, c'est-à-dire G.

Exercice n°1

Soit la fonction Inc suivante :

Fonction Inc(c :caractère, ch : chaine) :Entier

Début

P <-- -1

I <-- -1

Test <-- faux

Tant que i<long(ch)-1 et test=faux faire

I <-- i+1

Si ch(i)=c alors

Test <-- vrai

P <-- i

Fin si

Fin tant que

Retourner P

Fin

1. Compléter le tableau de déclaration des objets locaux :

objet	Type/nature

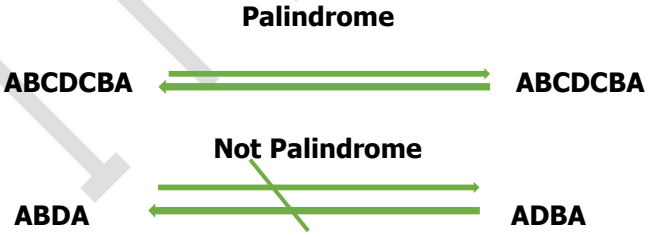
2. Donner les valeurs de i,P et test pour c="s"et ch="Dépression"

I						
Test						
P						

3. Déduire le rôle de la fonction Inc
4. Donner une fonction prédéfinie ayant le même rôle

Exercice n°2

Ecrire l'algorithme d'une fonction permettant de vérifier si une chaine de caractère et Palindrome ou non.



Exercice n°3

Chacune des 26 lettres est associée à l'un des entiers de 0 à 25, selon le tableau de correspondance suivant

A	B	C	D	E	F	G	H	I	J	K	L	M
0	1	2	3	4	5	6	7	8	9	10	11	12

N	O	P	Q	R	S	T	U	V	W	X	Y	Z
13	14	15	16	17	18	19	20	21	22	23	24	25

Le cryptage affine se fait à l'aide d'une clé, qui est un nombre entier K fixé, compris entre 0 et 25. Pour crypter une lettre donnée on suit le processus suivant :

- On repère le nombre x associé à la lettre dans le tableau de correspondance précèdent
- On multiplie ce nombre x pour la clé K
- On calcule le reste r de la division euclidienne du nombre obtenu par 26
- On repère la lettre associée au nombre r dans le tableau de correspondance, qui devient la lettre cryptée.

Par exemple, pour crypter la lettre P avec la clé k=11

- Le nombre x associé à lettre P et le nombre est 15
- On multiplie 15 par la clé 5, ce qui donne 11*15=165
- On calcule le reste de la division euclidienne par 26 : on obtient 165 MOD 26=9
- On repère finalement la lettre associée à 9 dans le tableau, c'est à dire J ainsi avec la clé k=11, la lettre P est cryptée en la lettre J.

<u>On crypte un mot en cryptant chacune des lettres de ce mot</u>

Exemple : pour une chaine "PYTHON" et k=14 la chaine cryptée est "GYGUOA"

On veut écrire un programme permettant de

- Saisir une chaine alphabétique majuscule ch.
- Saisir une clé k avec (1<=k<=25)
- Crypter la chaine ch suivant le principe décrit et afficher la chaine cryptée

1. Décomposer le programme en des modules
2. Ecrire les algorithmes des modules envisagés.

Exercice n°8 :
Soit A un 1er tableau rempli par n entiers aléatoires de l'intervalle [65,90] avec 10≤n≤3010

Écrire un algorithme d'un programme permettant de :

- Saisir n et remplir le tableau A.
- Remplir un 2ème tableau b à partir des éléments du tableau A, de telle sorte que chaque B[i]soit un caractère dont le code ASCII est égal à A[i].
- Afficher le tableau B.

Exemple :
Soit n=10

66	65	74	66	71	67	65	81	82	89

A

Le programme affiche:

B

B	A	J	B	G	C	B	Q	R	Y

Exercice 3 :

Un nombre K est dit **nombre heureux**, si on calcule la somme des carrés de chacun de ses chiffres, puis la somme des carrés des chiffres de ce résultat et ainsi de suite jusqu'à obtenir un nombre à un seul chiffre égal à 1.

Exemple :
K=7 est heureux, puisque :
7_2=49
42+92=974^2 + 9^2 = 9742+92=97
92+72=1309^2 + 7^2 = 13092+72=130
12+32+02=101^2 + 3^2 + 0^2 = 1012+32+02=10
12+02=11^2 + 0^2 = 112+02=1

On est arrivé à un nombre d'un seul chiffre qui est égal à 1, donc K=7K = 7K=7 est heureux.