Quelques fonctions et procédures

1. Saisie de données :

		Entier / Réel / Caractère / Chaîne de caractères	Tableau
Simple	**Algorithme**	Ecrire(″Saisir x″) Lire(x)	Pour i de binf à bsup faire Ecrire(″Saisir la case n°″,i) Lire(T[i]) Fin pour
	Python	x=input("Saisir x")	for i in range(binf,bsup+1): T[i]=input('Saisir la case n°'+str(i))
Contrôlée	**Algorithme**	Répéter Ecrire(″Saisir x″) Lire(x) Jusqu'à (condition)	Pour i de binf à bsup faire Répéter Ecrire(″Saisir la case n°″,i) Lire(T[i]) Jusqu'à (condition) Fin pour

	Python	x=input("Saisir x")
		while not(condition):
		x=input("Saisir x")
			for i in range(binf,bsup+1):
			x=input("Saisir x")
			while not(condition):
			x=input("Saisir x")

Saisir un entier x	Ecrire(″Saisir x″) Lire(x)	x=int(input("Saisir x"))

Saisir un entier x strictement positif	Procédure saisie(@ x :entier)
	Début
	Répéter
	Ecrire(″Saisir x″)
	Lire(x)
	Jusqu'à(x>0)
	Fin
		def saisie():
		x=int(input("Saisir x"))
		while not(x>0):
		x=input("Saisir x")
		return x

Saisir un entier négatif	Procédure saisie(@ x :entier)
	Début
	Répéter
	Ecrire(″Saisir x″)
	Lire(x)
	Jusqu'à(x<0)
	Fin
		def saisie():
		x=int(input("Saisir x"))
		while not(x<0):
		x=input("Saisir x")
		return x

Saisir un entier n (avec 2<n<50)	Procédure saisie(@ n :entier)
	Début
	Répéter
	Ecrire(″Saisir n″)
	Lire(n)
	Jusqu'à(2<n<50)
	Fin
		def saisie():
		n=int(input("Saisir n"))
		while not(2<n<50):
		n=input("Saisir n")
		return n

Saisir un entier n de trois chiffres	Procédure saisie(@ n :entier)
	Début
	Répéter
	Ecrire(″Saisir n″)
	Lire(n)
		def saisie():
		n=int(input("Saisir n"))
		while not(100<=n<=999):
		n=input("Saisir n")
		return n

	Jusqu'à(nϵ[100..999]) Fin	

Saisir un caractère c	Ecrire(″Saisir c″) Lire(c)	c=input("Saisir c")

Saisir une lettre minuscule	Procédure saisie(@ c :caractère)
	Début
	Répéter
	Ecrire(″Saisir c″)
	Lire(c)
	Jusqu'à(c ϵ ["a".."z"])
	Fin
		def saisie():
		c=input("Saisir c")
		while not("a"<=c<="z" and len(c)==1):
		c=input("Saisir c")
		return c

Saisir une lettre majuscule	Procédure saisie(@ c : caractère)
	Début
	Répéter
	Ecrire(″Saisir c″)
	Lire(c)
	Jusqu'à(c ϵ ["A".."Z"])
	Fin
		def saisie():
		c=input("Saisir c")
		while not("A"<=c<="Z" and len(c)==1):
		c=input("Saisir c")
		return c

Saisir un caractère numérique	Procédure saisie(@ c : caractère)
	Début
	Répéter
	Ecrire(″Saisir c″)
	Lire(c)
	Jusqu'à(c ϵ ["0".."9"])
	Fin
		def saisie():
		c=input("Saisir c")
		while not("0"<=c<="9" and len(c)==1):
		c=input("Saisir c")
		return c

Saisir un caractère 'O' ou 'N'	Procédure saisie(@ c : caractère)
	Début
	Répéter
	Ecrire(″Saisir c″)
	Lire(c)
	Jusqu'à(c ϵ {"O","N"})
	Fin
		def saisie():
		c=input("Saisir c")
		while not(c in {"O","N"}):
		c=input("Saisir c")
		return c

Saisir une chaîne de caractères	Ecrire(″Saisir ch″) Lire(ch)	ch=input("Saisir ch")

Saisir une chaîne de caractères dont la longueur ne dépasse pas 5	Procédure saisie(@ ch :chaîne)
	Début
	Répéter
	Ecrire(″Saisir ch″)
	Lire(ch)
	Jusqu'à(long(ch)<=5)
	Fin
		def saisie():
		ch=input("Saisir ch")
		while not(len(ch)<=5):
		ch=input("Saisir ch")
		return ch

Saisir une chaîne de caractères composée uniquement de lettres	Procédure saisie(@ ch :chaîne)
	Début
	Répéter
	Ecrire(″Saisir ch″)
	Lire(ch)
	Jusqu'à(alpha(ch))
		def saisie():
		ch=input("Saisir ch")
		while not(alpha(ch)):
		ch=input("Saisir ch")
		return ch

	Fin	

Saisir une chaîne de caractères composée uniquement de chiffres	Procédure saisie(@ ch :chaîne)
	Début
	Répéter
	Ecrire(″Saisir ch″)
	Lire(ch)
	Jusqu'à(Estnum(ch))
	Fin
		def saisie():
		ch=input("Saisir ch")
		while not(ch.isnumeric()):
		ch=input("Saisir ch")
		return ch

Saisir une chaîne de caractères composée uniquement des lettres 'A' et 'B'	Procédure saisie(@ ch :chaîne)
	Début
	Répéter
	Ecrire(″Saisir ch″)
	Lire(ch)
	Jusqu'à(AB(ch))
	Fin
		def saisie():
		ch=input("Saisir ch")
		while not(AB(ch)):
		ch=input("Saisir ch")
		return ch

Saisir un tableau de n éléments	Procédure remplir(@ T :tab,n :entier)
	Début
	Pour i de 0 à n-1 faire
	Ecrire(″Saisir la case n°″,i)
	Lire(T[i])
	Fin pour
	Fin
		from numpy import array
		T=array([str]*n)
		def remplir(T,n):
		for i in range(n):
		T[i]=input('Saisir la case n°'+str(i))

Saisir un tableau de n entiers positifs	Procédure remplir(@ T :tab,n :entier)
	Début
	Pour i de 0 à n-1 faire
	Répéter
	Ecrire(″Saisir la case n°″,i)
	Lire(T[i])
	Jusqu'à (T[i]>=0)
	Fin pour
	Fin
		from numpy import array
		T=array([int]*n)
		def remplir(T,n):
		for i in range(n):
		T[i]=int(input('Saisir la case n°'+str(i)))
		while not(T[i]>0):
		T[i]=int(input('Saisir la case n°'+str(i)))

Saisir un tableau de n lettres minuscules	Procédure remplir(@ T :tab,n :entier)
	Début
	Pour i de 0 à n-1 faire
	Répéter
	Ecrire(″Saisir la case n°″,i)
	Lire(T[i])
	Jusqu'à (T[i] ϵ ["a".."z"]))
	Fin pour
	Fin
		from numpy import array
		T=array([str()]*n)
		def remplir(T,n):
		for i in range(n):
		T[i]=input('Saisir la case n°'+str(i))
		while not(("a"<=T[i]<="z"):
		T[i]=input('Saisir la case n°'+str(i))

Saisir un tableau de n chaînes de caractères dont la longueur ne dépasse pas 6	Procédure remplir(@ T :tab,n :entier)
	Début
	Pour i de 0 à n-1 faire
	Répéter
	Ecrire(″Saisir la case n°″,i)
	Lire(T[i])
	Jusqu'à (long(T[i]) <=6)
	Fin pour
	Fin
		from numpy import array
		T=array([str]*n)
		def remplir(T,n):
		for i in range(n):
		T[i]=input('Saisir la case n°'+str(i))
		while not(len(T[i])<=6):
		T[i]=input('Saisir la case n°'+str(i))

	Procédure remplir(@ T :tab,n :entier) Début	from numpy import array T=array([str()]*n)

Saisir un tableau de n caractères 'A' et 'B'	Pour i de 0 à n-1 faire
	Répéter
	Ecrire(″Saisir la case n°″,i)
	Lire(T[i])
	Jusqu'à (T[i] ϵ{"A","B"})
	Fin pour
	Fin
		def remplir(T,n):
		for i in range(n):
		T[i]=input('Saisir la case n°'+str(i))
		while not(T[i]in{"A","B"}):
		T[i]=input('Saisir la case n°'+str(i))

Saisir un tableau de n éléments ordonnés	Procédure remplir(@ T :tab,n :entier)
	Début
	Ecrire(″Saisir la case n°0″)
	Lire(T[i])
	Pour i de 1 à n-1 faire
	Répéter
	Ecrire(″Saisir la case n°″,i)
	Lire(T[i])
	Jusqu'à (T[i] >T[i-1])
	Fin pour
	Fin
		from numpy import array
		T=array([str()]*n)
		def remplir(T,n):
		T[0]=int(input('Saisir la case n° 0')
		for i in range(1,n):
		T[i]=int(input('Saisir la case n°'+str(i)))
		while not(T[i]>T[i-1]):
		T[i]=int(input('Saisir la case n°'+str(i)))

Saisir un tableau de n éléments distincts	Procédure remplir(@ T :tab,n :entier)
	Début
	Pour i de 0 à n-1 faire
	Répéter
	Ecrire(″Saisir la case n°″,i)
	Lire(T[i])
	Jusqu'à (distinct(T,i))
	Fin pour
	Fin
		from numpy import array
		T=array([str()]*n)
		def remplir(T,n):
		for i in range(n):
		T[i]=input('Saisir la case n°'+str(i))
		while not(distinct(T,i)):
		T[i]=input('Saisir la case n°'+str(i))

2. Affichage de données :

		Entier / Réel / Caractère / Chaîne de caractères	**Tableau**
Simple	**Algorithme**	Ecrire(x)	Pour i de binf à bsup faire Ecrire(T[i]) Fin pour
	Python	print(x);	for i in range(taille): print(T[i])
Conditionné	**Algorithme**	Si(condition)alors Ecrire(x) Fin si	Pour i de binf à bsup faire Si(condition)alors Ecrire(T[i]) Fin si Fin pour

	Python	If(condition):
		print(x)
			for i in range(taille):
			if(condition):
			print(T[i])

Contrôlé	**Algorithme**	Répéter Ecrire(x) Jusqu'à (condition)	Pour i de binf à bsup faire Répéter Ecrire(T[i]) Jusqu'à (condition) Fin pour

	Python	While (condition):
		print(x)
			for i in range(taille):
			while (condition):
			print(T[i])

3. Modules usuels :

Somme des éléments entiers d'un tableau	fonction somme(t :tab ;n :entier) :entier
	Début
	  s 0
	  Pour i de 0 à n-1 faire
	  s s+t[i]
	  Fin pour
	  Retourner s
	Fin somme
		def somme(t,n):
		  s=0
		  for i in range(n):
		  s=s+t[i]
		  return s

Somme des éléments pairs d'un tableau	fonction somme(t :tab ;n :entier) :entier
	Début
	  s 0
	  Pour i de 0 à n-1 faire
	  Si t[i]mod2=0 alors
	  s s+t[i]
	  Fin si
	  Fin pour
	  Retourner s
	Fin
		def somme(t,n):
		  s=0
		  for i in range(n):
		  if i%2==0:
		  s=s+t[i]
		  return s

Somme des éléments caractères d'un tableau	fonction somme(t :tab ;n :entier) :chaîne
	Début
	  ch ""
	  Pour i de 0 à n-1 faire
	  ch ch+t[i]
	  Fin pour
	  Retourner ch
	Fin
		def somme(t,n):
		  ch=''
		  for i in range(n):
		  ch=ch+t[i]
		  return ch

Recherche de minimum d'un tableau	Fonction min(t :tab ;n :entier) :entier
	Début
	  m t[0]
	  Pour i de 1 à n-1 faire
	  Si(t[i]<m)alors
	  m t[i]
	  Fin si
	  Fin pour
	  Retourner m
	Fin
		def min(t,n):
		  m=t[0]
		  for i in range(1,n):
		  if t[i]<m:
		  m=t[i]
		  return m

Recherche de maximum d'un tableau	Fonction max(t :tab ;n :entier) :entier
	Début
	  m t[0]
	  Pour i de 1 à n-1 faire
	  Si(t[i]>m)alors
	  m t[i]
	  Fin si
	  Fin pour
	  Retourner m
	Fin
		def max(t,n):
		  m=t[0]
		  for i in range(1,n):
		  if t[i]>m:
		  m=t[i]
		  return m

Moyenne des éléments d'un tableau	Fonction moyenne(t :tab ;n :entier) :réel
	Début
	  s 0
	  Pour i de 1 à n faire
	  s s+t[i]
		def somme(t,n):
		  s=0
		  for i in range(n):
		  s=s+t[i]
		  return s/n

	Fin pour Retourner s/n Fin	

Recherche des occurrences d'un élément dans un tableau	Fonction occur(t :tab ;x,n :entier) :entier
	Début
	oc 0
	Pour i de 1 à n faire
	Si(t[i]=x)alors
	oc oc+1
	Fin pour
	Retourner oc
	Fin
		def occur(t,x,n):
		oc=0
		for i in range(n):
		if t[i]==x:
		oc=oc+1
		return oc

Conversion des éléments d'un tableau en une chaîne	Fonction convert(t :tab ;n :entier) :chaîne
	Début
	ch ″″
	Pour i de 1 à n faire
	ch ch+ convch(t[i])
	Fin pour
	Retourner ch
	Fin
		def convert(t,n):
		ch=""
		for i in range(n):
		ch=ch+str(t[i])
		return ch

Décalage des éléments d'un tableau vers la gauche	Procédure dec(@ t :tab ;p,f :entier)
	Début
	Pour i de p à f faire
	t[i] t[i+1]
	Fin pour
	Fin
		def dec(t,p,f):
		for i in range(p,f+1):
		t[i]=t[i+1]

Décalage des éléments d'un tableau vers la gauche avec suppression de case	Procédure dec(@ t :tab ;p,f :entier ;@n :entier)
	Début
	Pour i de p à f faire
	t[i] t[i+1]
	Fin pour
	n n-1
	Fin
		def dec(t,p,f):
		for i in range(p,f+1):
		t[i]=t[i+1]
		n=n-1
		return n

Décalage des éléments d'un tableau vers la droite	Procédure dec(@ t :tab ;p,f :entier)
	Début
	Pour i de f à p (pas=-1) faire
	t[i] t[i-1]
	Fin pour
	Fin
		def dec(t,p,f):
		for i in range(f,p+1,-1):
		t[i]=t[i-1]

Décalage des éléments d'un tableau vers la droite avec ajout de case	Procédure dec(@ t :tab ;p,f :entier ;n :entier)
	Début
	n n+1
	Pour i de f à p (pas=-1) faire
	t[i] t[i-1]
	Fin pour
	Fin
		def dec(t,p,f):
		n=n+1
		for i in range(f,p+1,-1):
		t[i]=t[i-1]
		return n

Somme des éléments numériques d'une chaîne	Fonction somme(ch :chaîne) :entier
	Début
	s 0
	Pour i de 0 à long(ch)-1 faire
	Si(ch[i]dans[″0″..″9″])alors
	s s+valeur(ch[i])
	Fin pour
		def somme(ch):
		s=0
		for i in range(len(ch))):
		if ch[i].isdigit():
		s=s+int(ch[i])
		return s

	Retourner s Fin	

Conversion des caractères d'une chaîne en un tableau	Procédure convers(ch :chaîne ;@ t :tab)
	Début
	Pour i de 0 à long(ch)-1 faire
	t[i] ch[i]
	Fin pour
	Fin
		def convers(ch):
		for i in range(len(ch)):
		t[i]ch[i]
		return t

Vérifier si un entier est premier	Fonction premier(x :entier) :booléen
	Début
	i 2
	Tant que(x mod i≠0)et(i<x) faire
	i i+1
	Fin tant que
	Premier i=x
	Fin
		def premier(x):
		i=2
		while x%i!=0 and i<x:
		i+=1
		return i==x

Décomposer un entier en facteurs premiers	Procédure decomp(@ t :tab ;x :entier ;@ n :entier)
	Début
	i 2
	n -1
	Répéter
	Si (x mod i=0)alors
	n n+1
	T[n] i
	x x div i
	Sinon
	i i+1
	Jusqu'à(x=1)
	Fin
		def decomp(x,t):
		i=2
		n=-1
		while x>1:
		if x%i==0:
		n+=1
		t[n]=i
		x=x//i
		else:
		i=i+1
		return n

Vérifier si une chaîne est palindrome	Fonction palindrome(ch :chaîne) :booléen
	Début
	i 0
	j long(ch)-1
	Tant que(ch[i]=ch[j])et (i<j) faire
	i i+1
	j j-1
	Fin tant que
	Retourner i>=j
	Fin
		def palindrome(ch):
		i=0
		j=len(ch)-1
		while ch[i]==ch[j] and i<j:
		i+=1
		j-=1
		return i>=j

Calculer x puissance n	Fonction puissance (x,n :entier) :entier Début p 1 Pour i de 1 à n faire p p*x Fin pour Retourner p Fin	def puissance(x,n) : p=1 for i in range(n) : p=p*x return p
Calculer la factorielle d'un entier	Fonction factorielle(x :entier) :entier Début f 1 Pour i de 2 à x faire f f*i Fin pour	def factorielle(x) : f=1 for i in range(1,x+1) : f=f*i return f

	Retourner f Fin	

Chercher les diviseurs d'un entier	Procédure diviseur(x :entier ;@ t :tab ;@ n :entier)
	Début
	n -1
	Pour i de 1 à x faire
	Si(x mod i=0)alors
	n n+1
	T[n] i
	Fin si
	Fin pour
	Fin
		def diviseur(x,t):
		n=-1
		for i in range(1,x+1):
		if x%i==0:
		n+=1
		t[n]=i
		return n

Calculer la somme des diviseurs d'un entier	Fonction sommediviseur(x :entier) :entier
	Début
	s 0
	Pour i de 1 à x faire
	Si(x mod i=0)alors
	s s+i
	Fin si
	Fin pour
	Retourner s
	Fin
		def sommediviseur(x):
		s=0
		for i in range(1,x):
		if x%i==0:
		s+=i
		return s

Chercher le nombre d'occurrences d'un caractère dans une chaîne	Fonction occurence(ch :chaîne ; c :caractère) :entier
	Début
	oc 0
	Pour i de 0 à long(ch)-1 faire
	Si(ch[i]=c)alors
	oc oc+1
	Fin si
	Fin pour
	Retourner oc
	Fin
		def occurence(ch,c) :
		oc=0;
		for i in range(len(ch)):
		if(ch[i]==c):
		oc=oc+1
		return oc

Trouver les caractères en commun de deux chaînes	Fonction commun(ch1,ch2 :chaîne):chaîne
	Début
	c ""
	Pour i de 0 à long(ch)-1 faire
	Si(pos(ch1[i],ch2)≠-1)alors
	c c+ch1[i]
	Fin si
	Fin pour
	Retourner c
	Fin
		def commun(ch1,ch2):
		c=''
		for i in range(len(ch1)):
		if(ch2.find(ch1[i])!=-1):
		c=c+ch1[i]
		return c

Trouver les caractères en commun de deux chaînes Sans redondance	Fonction commun(ch1,ch2 :chaîne):chaîne
	Début
	c ""
	Pour i de 0 à long(ch)-1 faire
	Si(pos(ch1[i],ch2)≠-1)et(pos(ch1[i],c)=-1)alors
	c c+ch1[i]
	Fin si
	Fin pour
	Retourner c
	Fin
		def commun(ch1,ch2):
		c=''
		for i in range(len(ch1)):
		if(ch2.find(ch1[i])!=-1)and(c.find(ch1[i])==-1):
		c=c+ch1[i]
		return c

Calculer le pgcd de deux entiers	Fonction PGCD(a,b :entier) :entier
	Début
	Tant que a≠b faire
	Si a>b Alors
	a ← a-b
	Sinon
	b ← b-a
	FinSi
	FinTantque
	Retourner a
	Fin
		def pgcd(a,b):
		while a!=b:
		if a>b: a=a-b
		else: b=b-a
		return a

Calculer le ppcm de deux entiers	Fonction ppcm(a,b :entier):entier
	Début
	max a
	min b
	Si b>a alors
	max b
	min a
	Fin si
	Tant que max mod min ≠0 faire
	max max+a+b –min
	Fin tant que
	Retourner max
	Fin
		def ppcm(a,b):
		mn=a
		mx=b
		if a>b:
		mn=b
		mx=a
		while mx%mn!=0:
		mx=mx+int(a)+int(b)-mn
		return mx

Vérifier si un entier est parfait	Fonction parfait(x :entier) :booléen
	Début
	Retourner x=sommediviseur(x)
	Fin
		def parfait(x):
		return x==sommediviseur(x)
		end;

Vérifier si une chaîne est pangramme (contient tous les lettres de l'alphabet)	Fonction pangramme(ch :chaîne) :booléen
	Début
	p vrai
	i "A"
	Répéter
	Si(pos(i,majus(ch))=-1)alors
	p faux
	Fin si
	i chr(ord(i)+1)
	Jusqu'à(i="Z")ou(p=faux)
	Retourner p
	Fin
		def pangramme(ch):
		i='A'
		p=True
		while i<'Z' and p:
		if(ch.upper().find(i)==-1):
		p=False
		i=chr(ord(i)+1)
		return p

Vérifier si deux chaînes sont anagrammes (se composent des mêmes caractères)	Fonction anagramme(ch1,ch2 :chaîne):booléen
	Début
	i 0
	Répéter
	Si(pos(ch1[i],ch2)≠-1)alors
	ch2 Efface(ch2,pos(ch1[i],ch2), pos(ch1[i],ch2)+1)
	ch1 Efface(ch1,i,i+1)
	Sinon
	i i+1
	Fin si
	Jusqu'à(ch2="")ou(pos(ch1[i],ch2)=-1)ou(ch1="")
	Retourner (ch2="")et(ch1="")
	Fin
		def anagramme(ch1,ch2):
		p= ch2.find(ch1[0])
		while ch2!="" and ch1!="" and p!=-1:
		if(p!=-1):
		ch2=ch2[:p]+ ch2[p+1:]
		ch1=ch1[1:]
		if ch1!="":
		p= ch2.find(ch1[0])
		return ch1=="" and ch2=""

Vérifier si une chaîne est tautogramme (tous les mots commencent par le même caractère)	Fonction tautogramme(ch :chaîne) :booléen
	Début
	  Répéter
	  Si(ch[pos(" ",ch)+1]=ch[0])alors
	  ch Efface(ch,pos(" ",ch), pos(" ",ch)+1)
	  Fin si
	  Jusqu'à(pos(" ",ch)=-1)ou(ch[pos(" ",ch)+1]≠ch[0])
	  Retourner pos(" ",ch)=-1
	Fin
		def tautogramme(ch):
		  p=ch.find(' ')
		  while p!=-1 and ch[p+1]==ch[0]:
		  ch=ch[:p]+ch[p+1:]
		  p=ch.find(' ')
		  return p==-1

Vérifier si une chaîne est composée uniquement par des lettres alphabétiques	Fonction alpha(ch :chaîne) :booléen
	Début
	  i 0
	  Test vrai
	  Tant que test=vrai et i<long(ch) faire
	  Si(majus(ch[i]dans["A".."Z"])alors
	  i i+1
	  Sinon
	  Test faux
	  Fin si
	  Retourner test
	Fin
		def alpha(ch):
		  i=0
		  test=True
		  while test and i<len(ch):
		  if "A"<=ch[i].upper()<="Z":
		  i+=1
		  else:
		  test=False
		  return test

Tri à bulles	Procédure tri_bulles(@ T :tab ;n :entier)
	Début
	  Répéter
	  Echange faux
	  Pour i de 1 à n-2 faire
	  Si T[i]>T[i+1] alors
	  X T[i]
	  T[i] T[i+1]
	  T[i+1] X
	  Echange vrai
	  Fin si
	  Fin pour
	  n n-1
	  Jusqu'à Echange=faux ou n=1
	Fin	

Tri par sélection	0- Def proc tri_selection(var T :tab ;n :entier) 1- Pour i de 1 à n-1 faire M i Pour j de i+1 à n faire Si T[M]>T[j] alors M j Fin si Fin pour X T[M] T[M] T[i] T[i] X Fin pour	

	2- Fin tri_selection	
Chercher un élément dans un tableau	0- Def Fn existe(T :tab ;n,x :entier):booléen 1- i 1 répéter Si(T[i]=x)alors test vrai Sinon test faux Fin si i i+1 jusqu'à i>=n ou test=faux 2- existe test 3- Fin existe	
Vérifier si les éléments d'un tableau sont distincts	0- Def Fn distinct(T :tab ;n :entier):booléen 1- i 1 Tant que i<n et n<>1 et T[i]<>T[n] faire i i+1 fin tant que 2- distinct i=n 3- Fin distinct	