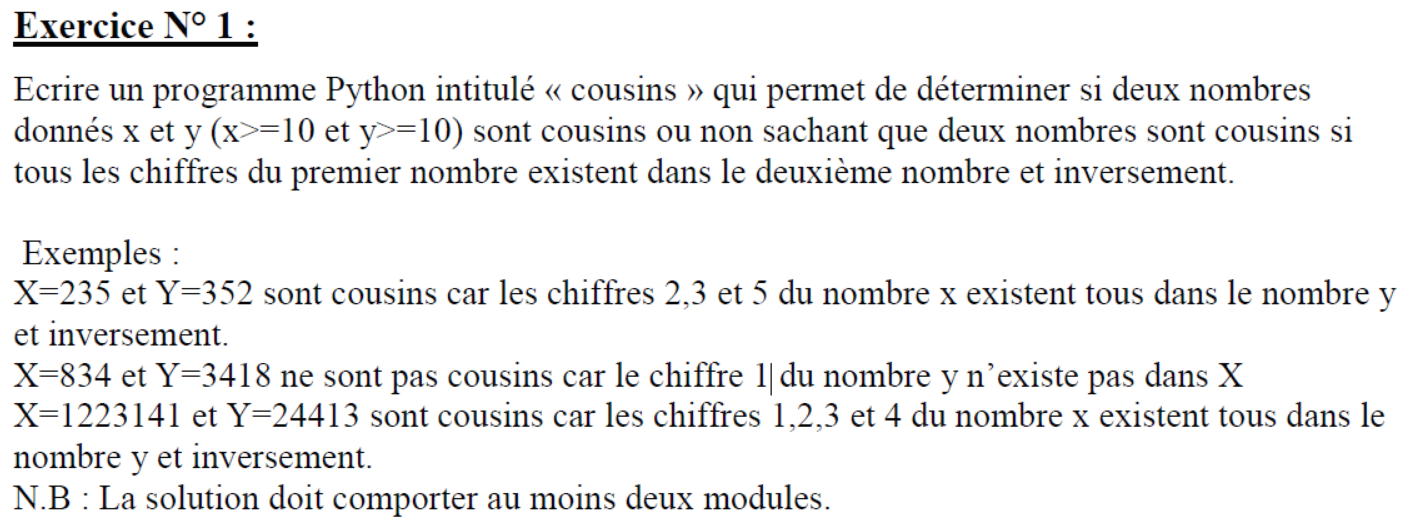
***Application :***



|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Algorithme cousins:   |  |  | | --- | --- | | TDO globaux : | | | objet | Type | | x,y | Entier | | Saisire | procedure | | affiche | procedure |   Saisire(x)  Saisire(y)  Affiche(x,y)  Fin  Saisie(@ a : entier)  Debut  Repéter  Lire(a)  Jusqu’à (a >= 10)  Fin  Affiche(x, y :entier)  Debut  Si verif(x, y) et verif (y,x) alors  Ecrire (‘cousins’)  Sinon  Ecrire (‘ne sont pas cousins’)  Fin si  Fin   |  |  | | --- | --- | | TDO locaux | | | objet | Type | | verif | fonction |   Fonction verif (x, y : entier)  : booléen  Ch1 <- conv ch (x)  Ch2 <- conv ch (y)  i <- 0  Tant que (i < long (ch1)) et (ch[i] C ch2) faire  i <- i +1  Fin tan que  v <- (i = long (ch1))  Retourner  Fin   |  |  | | --- | --- | | TDO locaux | | | i | Entier | | v | Booléen | | Ch1, ch2 | Chaine de caractère | | Pyhton :  def remplir ( ):  x = int(input("donner un entier : "))  while not ( x >= 10):  x = int(input("donner un entier : "))  return x  def verif (x,y):  v = True  ch1 = str(x)  ch2 = str(y)  i = 0  while (v == True) and (i < len(ch1)) :  if not ch1[i] in ch2 :  v = False  i = i + 1  return v  def affiche (x,y):  if verif(x,y) == True and verif(y,x) == True:  print("les deux nb sont cousins.")  else:  print("les deux nb ne sont pas cousins.")  x = remplir()  y = remplir()  affiche(x,y) |