```python
# Les instructions ci-dessous affichent les entiers de 1 à 10 (1 par ligne)
# On utilise la boucle while. Le programmeur doit gérer la variable i,
# en l'initialisant et en l'incrémentant d'une unité entre chaque tour de boucle.

i = 1
while i <= 10:
    print(i)
    i = i + 1

# Les instructions ci-dessous affichent les entiers de 1 à 10 (1 par ligne)
# On utilise la boucle for. Le programmeur n'a pas à gérer le fait que
# la variable i est initialisée à 1 et incrémentée à chaque tour de boucle :
# c'est fait automatiquement par l'instruction for.

for i in range(1, 11):
    print(i)

# Les instructions ci-dessous affichent les entiers 5, 10 et 15
# (1 par ligne), soit les entiers à partir de 5 jusqu'à 20 exclus
# de 5 en 5.

for i in range(5, 20, 5):
    print(i)

# La variable somme est initialisée à 0.
# La boucle for parcours les entiers de 1 à 10 exclus,
# de 1 en 1.
# A chaque itération, elle ajoute i à somme.
# Cette boucle calcule donc la somme des entiers de 1 à 9,
# soit 45.
somme = 0
for i in range(1, 10):
    somme = somme + i
print(somme)

# Même comportement que précédemment, écrit avec l'instruction while au lieu de
l'instruction for.
somme = 0
i = 1
while i <= 9:
    somme = somme + i
    i = i + 1
print(somme)

# Cette boucle est exécutée avec x allant de 1 à 10, par pas
# de 2.
# x prend donc les valeurs 1, 3, 5, 7 et 0.
# résultat est initialisé à 1 et à chaque tour de boucle,
# on le multiplie par x. resulat aura donc comme valeur finale
# 1 * 1 * 3 * 5 * 7 * 9, soit 945.
x = 1
resultat = 1
while x <= 10:
    resultat = resultat * x
    x = x + 2
print(resultat)

# Même résultat, mais avec une boucle for.
resultat = 1
for x in range(1, 11, 2):
    resultat = resultat * x
print(resultat)

# Le code ci-dessous affiche le résultat suivant (sans les ##) :
##0 Bonjour
##1 Bonjour
##2 Bonjour
##3 Bonjour
##4 Bonjour
```

```
69
70  # L'instruction for a été encapsulée dans une fonction, à laquelle on a donné
71  # un nom (ici : imprime_5), et que l'on peut réutiliser autant de fois que l'on
72  # souhaite, par imprime_5().
73
74  def imprime_5():
75      for i in range(0, 5):
76          print(i, "Bonjour")
77  imprime_5()
78
79
80  # Le code ci-dessous affiche le résultat suivant (sans les ##) :
81  ##0 Bonjour
82  ##1 Bonjour
83  ##2 Bonjour
84  ##3 Bonjour
85  ##4 Bonjour
86  ##5 Bonjour
87  ##6 Bonjour
88  ##7 Bonjour
89  ##8 Bonjour
90  ##9 Bonjour
91
92  # À nouveau, le code est encapsulé dans une fonction. La différence est qu'on a
93  # ajouté un paramètre, qui sera fixé lors de chaque utilisation de la fonction.
94  # Ici, par exemple, on a utilisé la fonction avec n valant 10 grâce à l'appel
imprime(10).
95  # On aurait pu aussi fixer la valeur de ce paramètre en écrivant imprime(n=10).
96
97  def imprime(n):
98      for i in range(0, n):
99          print(i, "Bonjour")
100 imprime(10)
101
102 # Le code ci-dessous affiche la table de multiplication de 7,
103 # de 0 jusqu'à 9.
104
105 def table_7():
106     for i in range(10):
107         print(i, "* 7 =", i*7)
108 table_7()
109
110 # Le fonction table(n) ci-dessous affiche la table de
111 # multiplication d'un entier n, de 0 jusqu'à 9.
112 # Le code suivant affiche les tables de 5 puis 7,
113 # en utilisant la fonction table(n).
114
115 def table(n):
116     for i in range(10):
117         print(i, "*", n, "=", i*n)
118
119 # Le code suivant affiche les tables de 5 puis 7,
120 # en utilisant la fonction table(n).
121
122 table(5)
123 table(7)
124
125 # Le fonction table_2(n, start, stop) ci-dessous affiche
126 # la table de multiplication d'un entier n, de start jusqu'à stop.
127
128 def table_2(n, start, stop):
129     for i in range(start, stop + 1):
130         print(i, "*", n, "=", i*n)
131
132 # Le code suivant affiche les tables de 5 de 3 à 8, puis de 7
133 # de 2 à 9.
134
135 table_2(5, 3, 8)
136 table_2(7, 2, 9)
```

```
137
138 # Le fonction table_3(n, m, start, stop) ci-dessous affiche
139 # les tables de multiplication des entiers de n à m, pour
140 # les valeurs allant de start jusqu'à stop.
141
142 def table_3(n, m, start, stop):
143     for i in range(n, m+1):
144         table_2(i, start, stop)
145
146 # Le code suivant affiche les tables de 5, 6 et 7, pour les
147 # valeurs allant de 3 à 8.
148
149 table_3(5, 7, 3, 8)
150
```