

# Horloge

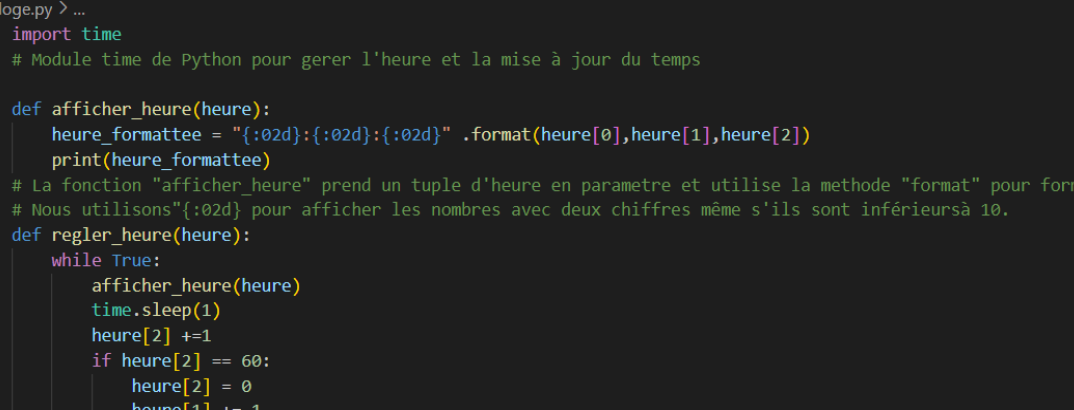
## Descriptif

Créez un programme qui affiche l'heure sous la forme suivante : **hh:mm:ss**. Par exemple, si l'heure est (16, 30, 0), le programme devra afficher 16:30:00.

Votre programme doit **actualiser** l'heure toutes **les secondes** jusqu'à l'arrêt de ce dernier.

Ajoutez une fonction nommée **"afficher\_heure"** qui permet de **régler l'heure**. Cette fonction devra prendre en paramètre une heure sous la forme d'un tuple (**heures**, **minutes**, **secondes**) et devra mettre à jour l'heure affichée grâce à la fonction **"afficher\_heure"**.

```
horloge.py x
horloge.py > regler_heure
1  import time
2  # Module time de Python pour gerer l'heure et la mise à jour du temps
3
4  def afficher_heure(heure):
5      heure_formattee = "{:02d}:{:02d}:{:02d}" .format(heure[0],heure[1],heure[2])
6      print(heure_formattee)
7  # La fonction "afficher_heure" prend un tuple d'heure en parametre et utilise la methode "format" pour forma
8  # Nous utilisons "{:02d}" pour afficher les nombres avec deux chiffres même s'ils sont inférieurs à 10.
9  def regler_heure(heure):
10     while True:
11         afficher_heure(heure)
12         time.sleep(1)
13         heure[2] +=1
14         if heure[2] == 60:
15             heure[2] = 0
16             heure[1] += 1
17             if heure[1] == 60:
18                 heure[1] = 0
19                 heure[0] += 1
20                 if heure[0] == 24:
21                     heure[0] = 0
22
```



The screenshot shows a code editor with a dark theme. The file 'horloge.py' is open, showing a Python script for a digital clock. The script imports the 'time' module and defines two functions: 'afficher\_heure' and 'regler\_heure'. The 'regler\_heure' function uses a 'while True' loop to update the time every second. The terminal at the bottom shows the output of the script, displaying the current time in HH:MM:SS format, incrementing by one second each line.

```
horloge.py X
horloge.py > ...
1 import time
2 # Module time de Python pour gerer l'heure et la mise à jour du temps
3
4 def afficher_heure(heure):
5     heure_formattee = "{:02d}:{:02d}:{:02d}" .format(heure[0],heure[1],heure[2])
6     print(heure_formattee)
7
8 # La fonction "afficher_heure" prend un tuple d'heure en parametre et utilise la methode "format" pour forma
9 # Nous utilisons "{:02d}" pour afficher les nombres avec deux chiffres même s'ils sont inférieurs à 10.
10 def regler_heure(heure):
11     while True:
12         afficher_heure(heure)
13         time.sleep(1)
14         heure[2] +=1
15         if heure[2] == 60:
16             heure[2] = 0
17             heure[1] += 1
18             if heure[1] == 60:
19                 heure[1] = 0
20                 heure[0] += 1
21                 if heure[0] == 24:
22                     heure[0] = 0
23
24 # La fonction "régler_heure" utilise une boucle "while True"pour mettre à jour l'heure toutes les secondes
25 # Timesleep(1)pour mettre le programme en pause pendant une seconde
26 # Ensuite nous incrémentons les secondes de l'heure 1 et vérifions si les minutes et les heures doivent etre
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Python + v [icon] [icon] [icon] [icon] [icon] [icon]

```
16:48:05
16:48:06
16:48:07
16:48:08
16:48:09
16:48:10
16:48:11
16:48:12
```

Ln 25, Col 117 Spaces: 4 UTF-8 CRLF Python Select Interpreter Go Live



Ajoutez une fonction qui permet de **régler l'alarme**. Cette fonction devra prendre en paramètre une heure sous la forme d'un tuple (**heures, minutes, secondes**) et devra afficher un message lorsque l'heure actuelle correspond à l'heure de l'alarme

```
def regler_alarme(alarme):  
    while True:  
        heure_actuelle = list(time())[3:6]  
        if tuple(heure_actuelle) == alarme:  
            print("Alarme ! L'heure actueele correspond à l'heure de l'alarme")  
            time.sleep(1)
```

```
26  
27 ~ def regler_alarme(alarme):  
28 ~     while True:  
29 ~         heure_actuelle = list(time())[3:6]  
30 ~         if tuple(heure_actuelle) == alarme:  
31 ~             print("Alarme ! L'heure actueele correspond à l'heure de l'alarme")  
32 ~             time.sleep(1)  
33  
34 # Exemple d'utilisation  
35 heure = [16,30,0]  
36 alarme = (17,0,0)  
37  
38 regler_heure(heure)  
39  
40
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Python + - [ ] [ ] ... ^ x

```
16:53:41  
16:53:42  
16:53:43  
16:53:44  
16:53:45  
16:53:46  
16:53:47  
16:53:48  
16:53:49  
16:53:50  
16:53:51  
16:53:52
```

## ... pour aller plus loin.

Ajoutez une fonction qui permet de **choisir** entre différents modes d'affichage de l'heure: 12 heures ou 24 heures. Si le mode 12 heures est sélectionné, l'heure devra être affichée sous la forme **hh:mm:ss AM/PM**. Par exemple, si l'heure est (16, 30, 0), elle devra être affichée comme 04:30:00 PM.



Mode affichage 12h

Mode affichage 24h

## ... pour aller encore plus loin.

Ajoutez une fonction qui permet de mettre en **pause** l'horloge. Cette fonction devra suspendre l'actualisation de l'heure jusqu'à ce qu'elle soit de nouveau relancée.

## Rendu

Le projet est à rendre sur <https://github.com/prenom-nom/horloge>.

Pensez à mettre vos repository en public !

## Compétences visées

- Maîtriser les bases de python
- Implémenter un algorithme

## Base de connaissances

- [La fonction sleep\(\)](#)
- [La fonction input\(\)](#)
- [La gestion du temps](#)