

```

##
# Corrigé des exercices "pour démarrer"
##

## Exo 1 (avec l'interpréteur)

#1)
reste = 4545646 % 121
quotient = 4545646 // 121

#2)
temps = 42569 # en secondes

# une première version
print("heures=" + str(secondes//3600))
print("minutes=" + str((secondes%3600)//60))
print("secondes=" + str(secondes%60))

# une deuxième version qui prend en charge aussi les jours et qui est automatisé avec
une fonction
temps = 42569 # en secondes
secondes = temps % 60
temps = temps // 60 # total de minutes
minutes = temps % 60
temps = temps // 60 # total d'heures
temps_affiche = str(minutes) + ' minutes, ' + str(secondes) + ' secondes'
if temps > 23:
    heures = temps % 24
    jours = temps // 24
    temps_affiche = str(jours) + ' jours, ' + str(heures) + ' heures, '+
temps_affiche
else:
    heures = temps
    temps_affiche = str(heures) + ' heures, '+ temps_affiche
print(temps_affiche)

def conversion(temps = 42569): # en secondes
    secondes = temps % 60
    temps = temps // 60 # total de minutes
    minutes = temps % 60
    temps = temps // 60 # total d'heures
    temps_affiche = str(minutes) + ' minutes, ' + str(secondes) + ' secondes'
    if temps > 23:
        heures = temps % 24
        jours = temps // 24
        temps_affiche = str(jours) + ' jours, ' + str(heures) + ' heures, '+
temps_affiche
    else:
        heures = temps
        temps_affiche = str(heures) + ' heures, '+ temps_affiche
    print(temps_affiche)

#3) La partie entière de -42,5 est -43
from math import floor
print(math.floor(-42.5))

#4)
a = 43.2e+3
print(type(a))
# a est de type flottant, c'est 43.2 *10^3

```

#5) Il affiche la chaîne de caractère 'yyy'

#6) on exécute `ord(' ')`, cela renvoie 32

Exercice 2

```
annee=2017
if (annee%4==0 and annee%100!=0) or annee%400==0:
    print(str(annee) + " est bissextile")
else:
    print(str(annee) + " n'est pas bissextile")
```

#rq: attention aux connecteurs logiques, le connecteur 'et' est prioritaire, donc
A and B or C est égal à (A and B) or C

Exercice 3

#1) cela affiche "supérieur à 2" et "supérieur à 4"
#2) cela affiche "supérieur à 2"
#3) cela affiche "supérieur à 4"

Exercice 4 (photocopies)

```
def prix(n):
    if n <= 20:
        return n*0.1
    else:
        return 2 + (n-20)*0.08
print("pour 15 : " + str(prix(15)) + " euros")
print("pour 30 : " + str(prix(30)) + " euros")
```

Exercice 5: multiplier c'est ajouter

```
s = 0
for k in range(1, 33): # pour k de 1 à 32
    s = s + 25
print(s)
#ou
s = 0
for k in range(32): # on répète 32 fois (pour k de 0 à 31)
    s = s + 25
print(s)
```

Exercice 6: Euler problem 1 " réponse: 233168

```
cible = 1000
s = 0
for k in range(cible):
    if k % 3 == 0 or k % 5 ==0:
        s = s + k
print(s)
```

Exercice 7:

#•Pour pouvoir réutiliser le résultat d'une fonction, celle-ci doit renvoyer quelque-chose. S'il n'y a pas de valeur de retour, l'objet renvoyé est de type Nonetype

```
# on corrige en tapant
def f(x):
    return 2*x + 1
```

Exercice 8

```
# après un return, on sort de la fonction!!
# truc(10) affiche 10 et renvoie 20
# bidule(10) affiche 10, puis 20 et renvoie 30
```

Exercice 9 (diviseurs d'un nombre triangulaire)

#1)

```
def triangle(n):
    s = 0
    for i in range(1,n+1):
        s = s + i
    return s
```

Bien sûr, on aurait pu écrire

```
def triangle(n):
    return n*(n+1)//2
```

#2)

```
def nbre_diviseurs_naif(n):
    """Entrée: n un entier naturel non nul
       Résultat: nombre de diviseurs de n
    """
    compteur = 1 # 1 déjà diviseur
    for d in range(2,n+1):
        if n % d == 0:
            compteur +=1
    return compteur
```

cette fonction effectue environ n divisions. Voici une deuxième version plus performante qui en effectue seulement racine de n environ. En effet, Les diviseurs marchent par couples mais attention au cas d'un carré: si $n = a^2$, le couple de diviseurs (a,a) ne fournit qu'un seul diviseur.

```
def nbre_diviseurs(n):
    """Entrée: n un entier naturel non nul
       Résultat: nombre de diviseurs de n
    """
    if n == 1:
        return 1
    compteur = 2 # 1 et n sont déjà diviseurs
    racine_n = math.floor(math.sqrt(n))
    for d in range(2,racine_n + 1):
        if n % d == 0:
            compteur +=2
    if math.sqrt(n) == math.floor(math.sqrt(n)):
        compteur -=1
    # si n est un carré, on a compté deux fois sqrt(n) comme diviseur
    return compteur
```

#3)

```
cible = 50
n = 1
while nbre_diviseurs(triangle(n)) <= cible:
    n = n+1
```

#Ceci est en réalité peu efficace car, pour calculer $\text{triangle}(n)$ à partir de $\text{triangle}(n-1)$, il suffit d'ajouter n . On préférera alors écrire:

```
cible = 50
n = 1
tn = 1
while nbre_diviseurs(tn) <= cible:
    n = n+1
    tn = tn + n
```

```

    tn = tn + n

## Ex0 10: fonction par morceaux

def f(x):
    if abs(x) <= 1:
        return x**2
    else:
        return 1

def g(x):
    if x > 0:
        return f(x)
    else:
        return -f(x)

## Exo 11: factorielle

def factorielle(n):
    p = 1 # p comme produit
    for k in range(1,n+1): # pour k de 1 à n
        p = p*k
    return p

## Exo 12: max de 2 ou 3 nombres

def max2(x, y):
    if x > y:
        return x
    else:
        return y

def max3(x,y,z):
    return max2(x, max2(y, z))

## Exo 13: exponentiation

#1)
p = 1
for k in range(13):
    p = p*21
print(p)

#2)
def puissance(x, n):
    p = 1
    for k in range(n):
        p = p*x
    return p

## Exo 14: calcul sur les rationnels

def somme_rationnels(r1, r2):
    """Données: r1 et r2 deux rationnels
    Renvoie le rationnel r1 + r2"""
    a = r1[0] # numérateur de r1
    b = r1[1] # dénominateur de r1
    c = r2[0] # numérateur de r2
    d = r2[1] # dénominateur de r2
    # r1 + r2 = a/b + c/d = (ad + bc)/ (bd)
    num = a*d + b*c
    denom = b*d
    return [num, denom]

# on remarque que cette fonction ne renvoie pas l'écriture irréductible de r1 + r2

```

```

# Exemple:
r1 = [3,5]
r2 = [1,2]

print(somme_rationnels(r1, r2))

def pgcd(a, b):
    """Fonction qui renvoie le pgcd des entiers a et b"""
    u = a
    v = b
    while v != 0: # on maintient invariant que pgcd(a,b) = pgcd(u,v)
        r = u % v
        u = v
        v = r
    return u

def somme_rationnels_irreduc(r1, r2):
    """Données: r1 et r2 deux rationnels
    Renvoie l'écriture irréductible du rationnel r1 + r2"""
    a = r1[0] # numérateur de r1
    b = r1[1] # dénominateur de r1
    c = r2[0] # numérateur de r2
    d = r2[1] # dénominateur de r2
    # r1 + r2 = a/b + c/d = (ad + bc)/ (bd)
    num = a*d + b*c
    denom = b*d
    d = pgcd(num, denom) # on va simplifier la fraction par d
    num = num//d
    denom = denom//d
    return [num, denom]

```