TP 04

Instructions conditionnelles et boucles

Sources :

Proposition de corrigé

Activité 1 : Bataille navale

Q 1 :

```
def touche(a,b,x):
    """Le navire se trouve entre les cases a et b, indique si le tir
    en x touche de navire"""
    xa,ya=a
    xb,yb=b
    xx,yx=x
    if xa==xb==xx :
        return (ya-yx)*(yb-yx) <= 0
    elif (yb==ya==yx) :
        return (xa-xx)*(xb-xx) <= 0
    else :
        return False
```

On pouvait aussi écrire le code suivant, plus astucieusement.

```
def touche(a,b,x):
    """Le navire se trouve entre les cases a et b, indique si le tir
    en x touche de navire"""
    return a <= x <= b or b <= x <= a
```

Activité 2 : Simulation d'un prêt immobilier

Q 2 :

```
def reste_a_payer(p,t,m,d):
    """p = montant du pret en euros
       t = taux mensuel
       m = mensualites
       d = duree en annees
       Calcule le montant restant a payer a l'echeance du pret"""
    dette = p
    for mois in range(d*12):
        # Inv : dette est dû au début du mois
        dette = dette*(1+t)-m
    return dette
```

Q 3 :

```
def somme_totale_payee(p,t,m,d):
    """p = montant du pret
       t = taux
       m = mensualites
       d = duree en annees
       Calcule le montant total paye"""
    return reste_a_payer(p,t,m,d) + 12*d*m
```

Q 4 :

```
def cout_total(p,t,m,d):
    """p = montant du pret
       t = taux
       m = mensualites
       d = duree en annees
       Calcule le cout total du credit"""
    return somme_totale_payee(p,t,m,d) - p
```

Q 5 :

```
def duree_mensualite(p,t,m):
    """Durée du prêt
       p = montant prêté
       t = taux mensuel
       m = mensualité"""
    emprunt = p
    d = 0
    while (1+t)*emprunt >= m:
        # Inv : emprunt est dû au début du mois d
        d = d+1
        emprunt = (1+t)*emprunt-m
    return d
```

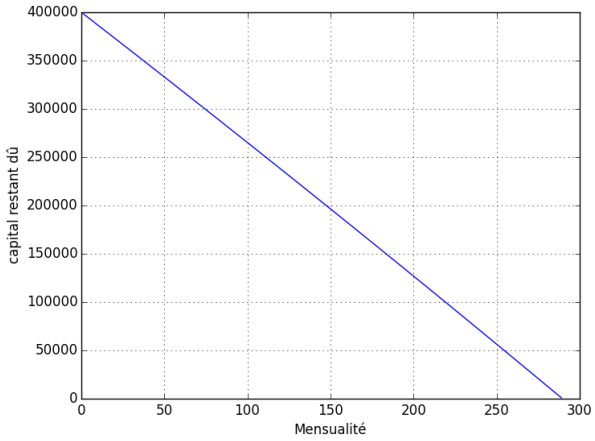
Q 6 :

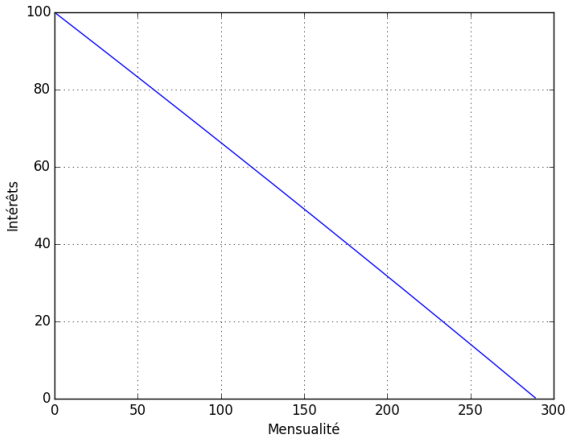
Si la mensualité est trop petite la dette augmentera plus vite que le capital restant du diminuera et ainsi la condition de la boucle conditionnelle ne sera jamais vérifiée et la boucle tournera à l'infini.

Q 7 :

```
import matplotlib.pyplot as plt
def tracer_mensualite(p,t,m):
    """Trace
       p = montant prêté
       t = taux mensuel
       m = mensualité"""
    emprunt = p
    d = 0
    mois=[]#numero de mensualite
    capital_rd=[]#capital restant du
    interets=[]
    while (1+t)*emprunt >= m:
        # Inv : emprunt est dû au début du mois d
        d = d+1
        emprunt = (1+t)*emprunt-m
        mois.append(d)
        capital_rd.append(emprunt)
        interets.append(t*emprunt)
    plt.clf()
    plt.plot(mois,capital_rd)
    plt.xlabel('Mensualité')
    plt.ylabel('capital restant du')
    plt.savefig('capital_restant_du.png')
```

```
    plt.clf()
    plt.plot(mois,interets,'b')
    plt.xlabel('Mensualité')
    plt.ylabel('Intérêts')
    plt.savefig('interets.png')
```





Activité 3 : Suites

Q 8 :

```
def f(n):
    """n = entier naturel
       Calcule le n-eme terme de la suite un"""
    u = 1
    v = 1
    for k in range(n):
        # Invariants : u = u_k et v = v_k
        u = .5*(u+(k+1)/u)
        v = v + (1/u**5)
        # u = u_(k+1) et v = v_(k+1)
    # Au dernier tour de boucle, k = n-1, donc v = v_(n-1+1)
    return v
```

Q 9 :
Il faut bien faire en sorte que les suive u_n et v_n soient calculées dans la même fonction et dans une même boucle.

Activité 4 : Sommes

Q 10 :

```
def somme1(n):
    """n = entier naturel.
       Calcule la somme des 1/(i+j**2), pour 1<=i,j<=n"""
    s = 0
    # Inv : s = somme des 1/(k+l**2) pour 1<=k<=0 et 1 <= l <= n (somme vide)
    for i in range(1,n+1):
        # Inv : s = somme des 1/(k+l**2) pour 1<=k<=(i-1) et 1 <= l <= n
        for j in range(1,n+1):
            # Inv : s = somme des 1/(k+l**2) pour 1<=k<=(i-1) et 1 <= l <= n
            #          + somme des 1 / (i + l**2) pour 1 <= l <= j-1
            s = s + 1 / (i + j**2)
            # Inv : s = somme des 1/(k+l**2) pour 1<=k<=(i-1) et 1 <= l <= n
```

```
            #          + somme des 1 / (i + l**2) pour 1 <= l <= j
        # Au dernier tour de boucle, j = n, donc
        # Inv : s = somme des 1/(k+l**2) pour 1<=k<=i et 1 <= l <= n
    # Au dernier tour de boucle, i = n, donc
    # s = somme des 1/(k+l**2) pour 1<=k<=n et 1 <= l <= n
    return s


def somme2(n):
    """n = entier naturel.
       Calcule la somme des 1/(i+j**2), pour 1<=i<j<=n"""
    s = 0
    # Inv : s = somme des 1/(k+l**2) pour 1<=k<=0 et k < l <= n (somme vide)
    for i in range(1,n+1):
        # Inv : s = somme des 1/(k+l**2) pour 1<=k<=i et i < l <= n
        for j in range(i+1,n+1):
            # Inv : s = somme des 1/(k+l**2) pour 1<=k<=i-1 et k < l <= n
            #          + somme des 1 / (i + l**2) pour i+1 < l <= j-1
            s = s + 1 / (i+j**2)
            # Inv : s = somme des 1/(k+l**2) pour 1<=k<=i-1 et k < l <= n
            #          + somme des 1 / (i + l**2) pour i+1 < l <= j
        # Au dernier tour de boucle, j = n, donc
        # Inv : s = somme des 1/(k+l**2) pour 1<=k<=i et k < l <= n
    # Au dernier tour de boucle, i = n, donc
    # s = somme des 1/(k+l**2) pour 1<=k<=n et k < l <= n
    return s
```