Programmes TD2 - Python

Exercice II.3 (nombres parfaits)

```
def somme_diviseurs(n):
    res=0
    for i in range(1,n+1):
        if (n%i==0):
            res=res+i
    return res


def est_parfait(n):
    s=somme_diviseurs(n)
    if (s==2*n):
        return 0
    elif (s>2*n):
        return 1
    else:
        return -1


n=int(input("Entrez un nombre"))
p=est_parfait(n)
if (p==0):
    print(n,"est parfait")
elif (p==1):
    print(n,"est abondant")
else:
    print(n,"est déficient")
```

Exercice II.4 (minimum de polynôme)

```
def polynome(x):
    return x**3-10*(x**2)-2610*x+84

mini = polynome(-50)
indice = -50

for i in range(-50,51):
    if polynome(i)<mini:
        mini=polynome(i)
        indice=i

print("La valeur minimale est",mini)
print("Elle est atteinte en",indice)
```

Fonction Syracuse

```
def Syracuse(x):
    if (x%2==0):
        return x/2
    else:
        return 3*x+1
```

Syracuse Question 1

```
N = int(input("Entrez un nombre :"))

u = N
while (u!=1):
    print(u)
    u=Syracuse(u)

print(u)
```

Syracuse Question 2 (avec une fonction)

```
def indice(N):
    u = N
    i = 0
    while (u!=1):
        u=Syracuse(u)
        i=i+1
    return i
```

Syracuse Question 3

```
for N in range(1,101):
    print("i(",N,") vaut :",indice(N))
```

Syracuse Question 4

```
maxi=indice(1)
posi=1

for N in range(1,101):
    if indice(N)>maxi:
        maxi=indice(N)
        posi=i

print("Le max vaut",maxi)
print("Il est atteint en",posi)
```

Syracuse Question 5 (avec une fonction)

```
def sommet(N):
    u = N
    max_atteint = N
    while (u!=1):
        u=Syracuse(u)
        if (max_atteint <u):
            max_atteint = u
    return max_atteint
```

Syracuse Question 6

```
maxi=sommet(1)
posi=1

for N in range(1,101):
    if sommet(N)>maxi:
        maxi=sommet(N)
        posi=i

print("Le max vaut",maxi)
print("Il est atteint en",posi)
```

II.6, question 1, n-ème nombre premier

```
from math import sqrt

# Détermine si n est premier
def premier(n):
    if (n==0 or n==1):
        return False
    else:
        p = 2
        while (p<=sqrt(n)):
            if (n%p==0):
                return False
            p=p+1
        return True


# Retourne le plus petit nombre premier
# strictement plus grand que p
def next_prem(p):
    x=p+1
    while not(premier(x)):
        x=x+1
    return x


# Retourne le n-ième nombre premier
def nieme_premier(n):
    p=2
    j=1
    while (j<n):
        p=next_prem(p)
        j=j+1
    return p
```

II.6, question 4, décomposition

```
# Retourne la puissance de p dans n
def puissance(n,p):
    c=0
    while (n%p==0):
        c+=1
        n=n/p
    return c
```

```
# Décompose n en produit de nombres premiers
def decompose(n):
    p=2
    while (p<=n):
        if (puissance(n,p)):
            print(p,"puissance",puissance(n,p))
            n=n/(p**puissance(n,p))
        p=next_prem(p)

decompose((2**2)*(3**3)*(5**2)*(7**2))
```