Exercice 1

1.

```
def MatNulle(n, p):
    M = []
    for i in range(n):
        L = []
        for j in range(p):
            L.append(0)
        M.append(L)
    return M
```

2.

```
def affiche(M):
    for L in M:
        print(L)
```

3.

```
def format(M):
    n = len(M)
    p = len(M[0])
    return (n, p)
```

4.

```
def MatIdentite(n):
    M = MatNulle(n, n)
    for i in range(n):
        M[i][i] = 1
    return M
```

5.

```
def mult(M, a):
    (n, p) = format(M)
    Q = MatNulle(n, p)
    for i in range(n):
        for j in range(p):
            Q[i][j] = a*M[i][j]
    return Q
```

6.

```
def somme(M, N):
    (n, p) = format(M)
    assert (n, p) == format(N), 'somme impossible, formats differents'
    Q = MatNulle(n, p)
    for i in range(n):
        for j in range(p):
            Q[i][j] = M[i][j] + N[i][j]
    return Q
```

7.

```
def prod(M, N):
    (n, p) = format(M)
    (q, r) = format(N)
    assert p == q, 'produit impossible ; les formats sont incompatibles'
    Q = MatNulle(n, r)
    for i in range(n):
        for j in range(r):
            S = 0
            for k in range(p):
                S = S + M[i][k]*N[k][j]
            Q[i][j] = S
    return Q
```

Exercice 2

1.

```
def degre(P):
    if P == []:
        return float('-inf') # le polynome nul a pour degre -infini
    else:
        return len(P) - 1
```

2.

```
def somme(P, Q):
    S = []
    if len(P) < len(Q):  # cas ou le degre de P est plus petit
        for i in range(len(P)):
            S.append(P[i] + Q[i])
        for i in range(len(P), len(Q)):
            S.append(Q[i])
    elif len(P) > len(Q):  # cas ou le degre de Q est plus petit
        for i in range(len(Q)):
            S.append(P[i] + Q[i])
        for i in range(len(Q), len(P)):
            S.append(P[i])
    else:   # Les degres sont egaux
        for i in range(len(P)):
            S.append(P[i] + Q[i])
        for i in range(len(P) - 1, -1, -1): # 'nettoyage' des coef. nuls
            if S[i] == 0:
                del S[i]
    return S
```

3.

```python
# solution courte mais complexe
def produit(P, Q):
    if P == [] or Q == []:
        return []
    Prod = []
    for k in range(degre(P) + degre(Q) + 1):
        S = 0
        for i in range(k + 1):
            if i <= degre(P) and 0 <= k - i <= degre(Q):
                S += P[i]*Q[k-i]
        Prod.append(S)
    return Prod
```

```python
# Pour la deuxieme solution on commence par definir le produit avec un monome

def prod_monome(P, n, a):
    """Renvoie le produit de P par a.x^n"""

    if P == []:
        return []

    Q = []
    for i in range(n):
        Q.append(0)
    for i in range(len(P)):
        Q.append(a*P[i])
    return Q

def produit2(P, Q):
    S = []
    for n in range(len(P)):
        S = somme(S, prod_monome(Q, n, P[n]))
    return S
```

4.

```python
def evaluation(P, x):  # Par l'algorithme de Horner
    S = 0
    for i in range(len(P) - 1, -1, -1):
        S = S*x + P[i]
    return S
```

5.

```python
def puissance(P, n): # algorithme naif
    Q = [1]
    for k in range(n):
        Q = produit(Q, P)
    return Q
```