```
#-------------------------------------------------
#----Answer Sheet: Polynomial Function Problem-----
#-------------------------------------------------

#Q1
def degree(p):
  return len(p)-1

#Q2
def affiche(p):
  s=str(p[0])
  for i in range(1,len(p)):
        if p[i]!=0:s+='+{}*x^{}'.format(p[i],i)
        # ou bien
        # if p[i]==0:continue
        # s+='+{}*x^{}'.format(p[i],i)
#Q3
def coeff(p,i):
  if i>=len(p) or i<0:return 0
  return p[i]

#Q4
def add(p1,p2):
  p=[]
  deg0=min(len(p1),len(p2))-1
  for i in range(deg0):
    p.append(p1[i]+p2[i])
  if len(p1)>len(p2)
    p.extend(p1[deg0:])
  if len(p2)>len(p1)
    p.extend(p2[deg0:])
  return p
# 2emme methode
  p1_=p1+max((len(p2)-len(p1)),0)*[0]
  p2_=p2+max((len(p1)-len(p2)),0)*[0]
# ou bien sachant que : k*<iterable> donne "l'iterable vide" du meme type.
#>>> 3* 'bjr'
#    bjrbjrbjr
#>>> -3* 'bjr'
#    ''
#>>> 3* [0]
#    [0,0,0]
#>>> -3* [0]
#    []

  p1_=p1+(len(p2)-len(p1),0)*[0]
  p2_=p2+(len(p1)-len(p2),0)*[0]
  return [p1_[i]+p2_[i] for i in range(len(p1_)) ]

#Q6
def multint(P,N):
  #return mult(P,[N])
  return [N*e for e in P]

#Q7
def sub(p1,p2):
  #return add(p1,multint(p2,-1))
  return add(p1,mult(p2,[-1]))

#Q8
def primitive(p):
  return[0]+[p[i]/(i+1) for i in range(len(p))]

# Q9
def deriv(p)
  return [p[i]*i for i in range(1,len(p))]
```