DS1 2024-2025 Corrigé

In [ ]:
```
# Q1
z = a**2/(a+b*sin(x)) + b/3/x**2
```

Q2

a/b donne un flottant

a//b donne un entier

Q3

5//2 vaut 2, quotient de la division entière, entier (pas 2. qui est un flottant)

5%2 vaut 1, reste de la division entière, entier (pas 1. qui est un flottant)

Q4

affiche l'entier 3, partie entière du nombre décimal 3.5

Q5

5 valeurs affichées, la dernière étant 16.

Q6

Il y a une erreur dans le script qui doit commencer par : `x = 0`

x est incrémenté de 1 à chaque passage. La boucle s'arrête lorsque la condition n'est plus valide, soit pour x valant 10.

Q7

les entiers impairs de 0 à 9

Q8

a vaut 3 et b vaut 4

Q9

h(5) retourne 'B'

h(10) retourne 'C'

h(1) retourne 'A'

Q10

Après le script 1, res vaut None (pas d'objet retourné spécifiquement par la fonction, donc c'est l'objet par défaut None qui est retourné).

Après le script 2 : res vaut le couple (1., 0.)

Après le script 3 : res vaut None, valeur par défaut.

Seul le script 2 permet d'évaluer l'expression `f(1)*2` . f(1) étant un n-uplet, la multiplication par 2 est une répétition. Le résultat est (1., 0., 1., 0.)

In [ ]:
```
# Q11
def max(a, b, c):
    #a = 1 -> respect des paramètres transmis par l'utilisateur
    if a > b: # manque :
        if a > c:
            m = a
        else:
            m = c
    else:
        if b > c: # comparaison inversée, ou il faut permuter b et c ci-dessous
            m = b
        else:
            m = c
    return m  # pb indentation et confusion print/return
```

In [ ]:
```
# Q12
def S(n):
    s = 0
    for i in range(1, n+1):
        for j in range(i, n+1):
            s = s + i / j
    return s
```

In [1]:
```
# Q13
txt = "PCSI"
txt[-1] + txt[0] + txt
```

Out[1]: 'IPPCSI'

In [3]:
```
# Q14
(txt[2] + txt[3])*2
```

Out[3]: 'SISI'

In [ ]:
```
# Q15
def c(i, a):
    ci = 1
    for k in range(1, i+1):
        ci = ci * (a + 1 - k) / k
```

```
    return ci

# Q16
def u(x, a, n):
    un = 1
    for i in range(1, n+1):
        un = un + c(i, a) * x**i
    return un
```

In [5]:
```
# Q17
def maximum(f):
    max = f[0]
    for fx in f: # parcours par élément
        if fx > max:
            max = fx
    return max

f = [2, 3, 1, 1]
print(maximum(f))

def maximum(f):
    max = f[0]
    n = len(f)
    for x in range(n): # parcours par indice
        if f[x] > max:
            max = f[x]
    return max

print(maximum(f))
```

```
3
3
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [7]:
```
# Q18
```