1

```
# Ex 1
b = [0,0,4,0,0,2,7,0]

A = []
for i in range(0,N):
    row = []
    for j in range(0,N):
        if i==j or j==N-i-1:
            el = b[i]
        else:
            el = 0
        row += [el]
    A += [row]

for i in range(0,N):
    s = 0
    for j in range(0,N):
        s += ( A[i][j] - A[j][i] ) * b[j]
    c += [s]
```

- set CID into a 1D array [1]
- set matrix A: two loops to capture 2D nature [1]
  main diagonal ok [2]
  anti-diagonal ok [2]
- correct identification of transpose [2]
- correct identification of difference A-AT [2]
- correct product [2]
- all in one loop [2]

```
# set s1, s2
s1 = 0/10+0.9
s2 = 3/10+0.6

# start moving
while yc >=0:
    dx = rd.random()*(s2+s1)-s1
    dy = rd.random()*(s2+s1)-s1
    xn = xc + dx
    yn = yc + dy
    rs = xn**2 + yn**2
    if 100<rs<400:
        pl.plot([xc,xn],[yc,yn],c='r')
        xc = xn
        yc = yn
```

- plotting the doughnut, by any means [3]: attempt 1 + correct 2
- correct generation of boundaries s1 and s2 [4]: dx, dy independent 2 + correct 2
- setting while loop (unconditional loop) [2]
- correct condition on looping [2]
- make advancement in both dx and dy [2]
- condition on hitting the walls [2]
- bouncing back when hitting the wall [2]
- plotting [3]: attempt 1 + correct 2

```
day = []
infect = []
for i in range(0,N):
    if i%15 == 0:
        # start a new week
        c = 1
    else:
        if c%2 == 1:
            day += [t[i].rstrip()]
        else:
            infect += [int(t[i])]
        c += 1

# analyse data
# overall number of infections
tot = 0
days2000 = []
infectw = []
totw = 0
for i in range(0,Nd):
    # overall infections
    tot += infect[i]
    # days with high infections (>2000)
    if infect[i]>2000:
        days2000 += [day[i]]

    # compute weekly number of infections
    # add this number of infections to the weekly total
    totw += infect[i]
    # check if end of week
    if (i+1)%7 == 0:
        # store the weekly total
        infectw += [totw]
        # reset the weekly total
        totw = 0

Nw = len(infectw)
ratew = []
maxw = infectw[0]
week = ['1']
for i in range(1,Nw):
    if infectw[i] > maxw:
        maxw = infectw[i]
        week = str(i+1)
```

- reading data correctly: file reading, stripping trails, converting data into numbers [2]
- sorting the data from serial into a database like [2]
- overall number of infections [1] 273319
- weekly number of infection [5]: attempt 1 + logic makes some sense 2 + correct 2

[241, 1118, 3623, 12075, 24814, 37088, 35226, 34160, 33883, 33000, 24901, 17518, 15672]

- list of day with > 2000 [3]: attempt 1 + correct 2

['2020-03-27', '2020-03-28', '2020-03-29' …… '2020-05-25', '2020-05-27', '2020-05-28', '2020-05-30']

- weekly increment [5]: attempt 1 + logic makes some sense 2 + correct 2 [363.90, 224.06, 233.2 ….-24.54, -29.64, -10.53]
- week with highest number of infection [3]: attempt 1 + correct 2 6

`    ratew += [(infectw[i]-infectw[i-1])/infectw[i-1]*100]`	

`def fact(n):`
`    if n == 0:`
`        res = 1`
`    else:`
`        res = n * fact(n-1)`
`    return res`
 	
`def sum(i):`
`    if i == 1:`
`        res = 1`
`    else:`
`        res = sum(i-1) + i**4/fact(i-1)`
`    return res`
	- correct definition of function sum [1]
	- correct use of ending condition [2]
	- correct use of recursive formula [2]
	- correct computation of factorial, even if not with function [2]
	
	Or
	
	- if iterative instead of recursive [4]: make sense 2 + correct 2

Comments	[3]: exhaustive and relevant - [2]: present, but not so clear and meaningful – [1]: scarce

Benchmarking

Task 1

```
A = [[1,2,3,4,5,6],[6,0,3,4,2,1],[2,1,9,4,6,7],[3,0,8,4,6,5],[1,5,4,3,3,1],[6,1,4,4,3,2]]
for row in A:
    print(row)
print()
T = Decompose(A)
print()
for row in T:
    print(row)
```

```
[0, 0, 27, 16, 15, 12]
[6, 0, 27, 16, 6, 2]
[2, 0, 0, 16, 18, 14]
[3, 0, 72, 0, 18, 10]
[1, 0, 36, 12, 0, 2]
[6, 0, 36, 16, 9, 0]
```

Task 2

Try a positive and a negative angle and inspect the outcome

Task 3