Lösningar och kommentarer för tentamen 2026-08-17 i DA2005 (och DA2004)

Del A: flervalsfrågor

1. C

2. B

3. A

4. D

5. B

6. A

7. B, C, E

8. C, D

Del B: kodfrågor

9. Exempellösning:

A. Funktionen förutsätter att indata är något som kan itereras över, som en lista, med tal som element. Den itererar igenom varje tal och beräknar elementets värde modulo 2. Om det är noll, dvs talet är jämnt, så läggs *kvadraten av talet* till sist i listan *result*. Slutligen returneras resultatlistan. Kort sagt: funktionen returnerar en lista med kvadraten av de tal i indata som är jämna.

B. `[16, 4, 64]`

10.

- Filen öppnas inte: man behöver skriva `with open(filename) as f:` för att det ska fungera.
- Variabeln `n` initieras inte.

11.

```
try:
    result = 100 / temperature
    print(result)
except ZeroDivisionError:
    print('Temperature cannot be zero')
except:
    print('An error occured')
```

12.

```
class FlightClass:
    def __init__(self, flight_number, origin, destination):
        self.flight_number = flight_number
        self.origin = origin
        self.destination = destination
        self.status = 'upcoming'

        def set_as_landed(self):  # Question 12B
            self.status = 'landed'

# Question 12A
flight = FlightClass('SK2026', 'ARN', 'UME')
flight.status = 'in the air'

# Question 12B
flight.set_as_landed()
```

13.

```
# A och B
def header_level(line):
    if not line:
        return 0

    n_levels = 0
    while line[n_levels] == '#':
        n_levels += 1
    if n_levels > 3:
```

```
            raise Exception('Too many header levels')
        else:
            return n_levels

    # C
    def table_of_contents(filename):
        with open(filename) as h:
            for line in h:
                line = line.strip() # Finess: strip tar bort onödiga radbrytningar
                if header_level(line) == 1:
                    print(line[2:])
                elif header_level(line) == 2:
                    print('  ', line[3:])
```

Del C: kodfrågor inspirerade av labbarna

14.
```
import math
def inside_unit_circle(x, y):
    radius = math.sqrt(x*x + y*y)
    return radius < 1.0
```

Som flera insåg i tentasalen så behövs inte math.sqrt för att avgöra om man är innanför enhetscirkeln, men frågan kräver tydligt att man ska kunna beräkna roten.

15.
```
def deci_to_hexa(num):
    '''
    Convert positive integers to a hexadecimal string representation
    '''
    base = 16
    lookup = ['0', '1', '2', '3', '4', '5', '6', '7', '8', '9', 'A', 'B', 'C', 'D', 'E', 'F'
        ]
    if num == 0:
        return '0'

    hex_result = ''
    while num > 0:
        digit = lookup[num % base]
        hex_result = digit + hex_result
        num = num // base
    return hex_result
```

16.A. Här är två olika lösningar. Den andra är snabbare än den första (varför?), och mer trogen till algoritmtexten.

```
def bubble_sort1(lst):
    for last_idx in range(len(lst)-1, -1, -1):
        for idx in range(last_idx):
            if lst[idx] > lst[idx+1]:
                tmp = lst[idx+1]
                lst[idx+1] = lst[idx]
                lst[idx] = tmp
    return lst # Feels weird not to return something

def bubble_sort2(lst):
    still_swapping = True
    while still_swapping:
        still_swapping = False  # A hypothesis
        for idx in range(len(lst)-1):
            if lst[idx] > lst[idx+1]:
                tmp = lst[idx+1]
                lst[idx+1] = lst[idx]
                lst[idx] = tmp
                still_swapping = True # Hypothesis was false
    return lst # Feels weird not to return something
```

B. Elementen kommer i omvänd ordning, tex: 4, 3, 2, 1.

C. `assert bubble_sort([4,3,2,1])== [1,2,3,4]`

17.
```
import math
from  matplotlib.pyplot import plot, savefig, show

def plot_my_fcn(fcn, x1=-5, x2=5):
    xs = []
    ys = []
    for x in range(x1, x2+1):
        xs.append(x)
        ys.append(fcn(x))
    fig = plot(xs, ys)
    savefig('my_fcn.pdf')
```