Facit och kommentarer till prov 2021-02-03 i DA2004/DA2005

Del A: flervalsfrågor

1. *C*

2. *A,B,E*

3. *B,C*

4. *C*

5. *B,C,D*

6. *C*

7. *A*

8. *C*

Del B: kodfrågor

9. Möjliga lösningar:

```
# option 1
def flip_for(s):
    s_str = ''
    for i in range(len(s)-1, -1, -1):
        s_str += str(s[i])
    return s_str

# or, option 2
def flip_for(s):
    s_str = ''
    for i in s:
        s_str += str(i)
    return s_str[::-1]
```

10. Möjlig lösning:

```
def dna_sum(s):
    if not isinstance(s, str):
        raise ValueError(
            "<dna_sum: argument must be a string>")
    d = {'A': 1, 'C': 2, 'G': 3, 'T': 4}
    dna_sum = 0
    for c in s:
        dna_sum += d.get(c, 0)
    return dna_sum
```

11. Möjlig lösning:

```
def dna_sum_recursive(s):
    if not isinstance(s, str):
        raise ValueError(
            "<dna_sum: argument must be a string>")
    d = {'A': 1, 'C': 2, 'G': 3, 'T': 4}
    if len(s) == 0:
        return 0
    else:
        return d.get(s[0], 0) + dna_sum_recursive(s[1:])
```

12. Möjlig lösning:

```
def min_index(list_in):
    min_val = 2**100
    min_ind = -1
    for i in range(len(list_in)):
        if hash(list_in[i]) < min_val:
            min_ind = i
            min_val = hash(list_in[i])
    return min_ind
```

eller

```
def min_index2(list_in):
    if not list_in:
        return -1
    return min([(hash(list_in[i]), i) for i in range(len(
        list_in))])[1]
```

13. Möjlig lösning:

```
class Vehicle(object):
    """Abstract class for vehicle
    """
    def __init__(self, n_wheels, max_velocity):
        self.max_velocity = max_velocity
        self.n_wheels = n_wheels
        self.helmet_needed = False if n_wheels > 2 else True
        self.vehicle_type = 'Vehicle'

    def __str__(self):
        return (f"<{self.vehicle_type} | Helmet needed: {self.
            helmet_needed}, "
                f"Max velocity: {self.max_velocity} km/h>")
```

14. Möjlig lösning:

```
def select_data(fn, filter_float):
    with open(fn, 'r') as fp, open('output.txt', 'w') as fp2:
        filter_data = map(
            lambda x: ','.join(x),
            filter(lambda x: float(x[1]) > filter_float,
                   map(lambda x: x.split(','), fp)))
        fp2.writelines(filter_data)
```