

# Løsningsforslag

## Runde 2

### 2.5 Programvaretesting

#### 2.5.1

Bruk testdrevet utvikling til å lage funksjonen `bmi_klassifisering(høyde:float, vekt:float) -> str`. Denne funksjonen skal returnere klassifiseringen basert på høyde og vekt. BMI beregnes som  $\text{vekt}/(\text{høyde}*\text{høyde})$ , hvor høyden er oppgitt i **meter** og vekten i **kg**. Argumentene skal være desimaltall med to desimaler for høyden og én desimal for vekten. Vi antar at argumentene allerede er kontrollert for å være realistiske og av riktig type før de brukes i funksjonen, så vi trenger ikke å teste dette i oppgaven. Vennligst lever hele mappen med alle filene i en zippet fil.

Klassifisering	BMI
Undervekt	18,4 eller lavere
Normalvekt	18,5-24,9
Overvekt	25,0-29,9
Fedme, grad 1	30-34,9
Fedme, grad 2	35-39,9
Fedme, grad 3	40,0 eller høyere

```

s_2_5_1_bmi.py X test_s_2_5_1_bmi.py
s_2_5_1_bmi.py > bmi_klassifisering
1 def bmi_klassifisering(høyde:float,vekt:float)->str:
2     bmi = vekt/høyde**2
3
4     if bmi<18.5: return 'Undervekt'
5     elif bmi<25 : return 'Normalvekt'
6     elif bmi<30 : return 'Overvekt'
7     elif bmi<35 : return 'Fedme, grad 1'
8     elif bmi<40 : return 'Fedme, grad 2'
9     else:       return 'Fedme, grad 3'

```

```

File Edit Selection View Go Run Terminal Help
s_2_5_1_bmi.py test_s_2_5_1_bmi.py X
test_s_2_5_1_bmi.py > test_overvekt
1 from s_2_5_1_bmi import bmi_klassifisering
2
3 def test_undervekt():
4     assert bmi_klassifisering(1.80,48.6) == 'Undervekt'
5     assert bmi_klassifisering(1.80,59.9) == 'Undervekt'
6     assert bmi_klassifisering(1.75,55.0) == 'Undervekt'
7
8 def test_normalvekt():
9     assert bmi_klassifisering(1.80,60.0) == 'Normalvekt'
10    assert bmi_klassifisering(1.80,75.0) == 'Normalvekt'
11    assert bmi_klassifisering(1.75,76.5) == 'Normalvekt'
12
13 def test_overvekt():
14    assert bmi_klassifisering(1.72,74.0) == 'Overvekt'
15    assert bmi_klassifisering(1.72,79.8) == 'Overvekt'
16    assert bmi_klassifisering(1.85,102.6) == 'Overvekt'
17
18 def test_fedme_1():
19    assert bmi_klassifisering(1.50,67.5) == 'Fedme, grad 1'
20    assert bmi_klassifisering(1.50,72.5) == 'Fedme, grad 1'
21    assert bmi_klassifisering(1.75,107.1) == 'Fedme, grad 1'
22
23 def test_fedme_2():
24    assert bmi_klassifisering(1.80,113.4) == 'Fedme, grad 2'
25    assert bmi_klassifisering(1.80,119.0) == 'Fedme, grad 2'
26    assert bmi_klassifisering(1.75,108.5) == 'Fedme, grad 2'
27
28 def test_fedme_3():
29    assert bmi_klassifisering(1.80,160.0) == 'Fedme, grad 3'
30    assert bmi_klassifisering(1.80,152.3) == 'Fedme, grad 3'
31    assert bmi_klassifisering(1.75,124.0) == 'Fedme, grad 3'

```

Se vedlagte mappe 2\_5\_1\_bmi

## 2.5.2

Vi ønsker å lage et program der brukeren skal skrive inn en poengsum fra og med 0 til og med 100, og programmet skal bestemme og skrive ut tilsvarende karakter etter følgende regler:

- Enhver poengsum under 50 får karakteren «Ikke bestått».
- Enhver poengsum fra og med 50 til og med 69 får karakteren «Bestått».
- Enhver poengsum fra og med 70 til og med 89 får karakteren «Godt bestått».
- Enhver poengsum på 90 eller høyere får karakteren «Meget godt bestått».
- Enhver poengsum som er mindre enn 0 eller større enn 100, får karakteren «Ikke gyldig poengsum!».

Bruk den foreslåtte funksjonen nedenfor, gjennomfør testene som er beskrevet i den følgende testplanen, og fyll inn de faktiske resultatene du får. Ta tiden du bruker for hver deloppgave.

- Ved hjelp av `print`-setninger
- Ved hjelp av `assert`-setninger
- Ved hjelp av `pytest` med parametere

```
def karakter(poengsum:int)->str:
    if poengsum < 50:
        return('Ikke bestått')
    elif 50 < poengsum < 69:
        return('Bestått')
    elif 70 < poengsum < 89:
        return('Godt bestått')
    elif 90 < poengsum < 100:
        return('Meget godt bestått')
    else:
        return('Ikke gyldig poengsum!')
```

Test nr.	Input-verdier	Forventet resultat	Faktisk resultat
1	Poengsum: 30	Ikke bestått	
2	Poengsum: 65	Bestått	
3	Poengsum: 82	Godt bestått	
4	Poengsum: 97	Meget godt bestått	
5	Poengsum: 102	Ikke gyldig poengsum!	
6	Poengsum: 0	Ikke bestått	
7	Poengsum: 50	Bestått	
8	Poengsum: 69	Bestått	
9	Poengsum: 70	Godt bestått	
10	Poengsum: 89	Godt bestått	
11	Poengsum: 90	Meget godt bestått	
12	Poengsum: 100	Meget godt bestått	
13	Poengsum: -1	Ikke gyldig poengsum!	

Se filene `karakter_modul.py` og `testresultater.xlsx`.

Vennligst lever hele mappen med alle filene i en zippet fil.

- Ved hjelp av `print`-setninger

Test nr.	Input-verdier	Forventet resultat	Faktisk resultat
1	Poengsum: 30	Ikke bestått	Ikke bestått
2	Poengsum: 65	Bestått	Bestått
3	Poengsum: 82	Godt bestått	Godt bestått
4	Poengsum: 97	Meget godt bestått	Meget godt bestått
5	Poengsum: 102	Ikke gyldig poengsum!	Ikke gyldig poengsum!
6	Poengsum: 0	Ikke bestått	Ikke bestått
7	Poengsum: 50	Bestått	Ikke gyldig poengsum!
8	Poengsum: 69	Bestått	Ikke gyldig poengsum!
9	Poengsum: 70	Godt bestått	Ikke gyldig poengsum!
10	Poengsum: 89	Godt bestått	Ikke gyldig poengsum!
11	Poengsum: 90	Meget godt bestått	Ikke gyldig poengsum!
12	Poengsum: 100	Meget godt bestått	Ikke gyldig poengsum!
13	Poengsum: -1	Ikke gyldig poengsum!	Ikke bestått

Se vedlagte `print_setninger.py` i mappen `2_5_2_karakter`

## Smidig IT-2

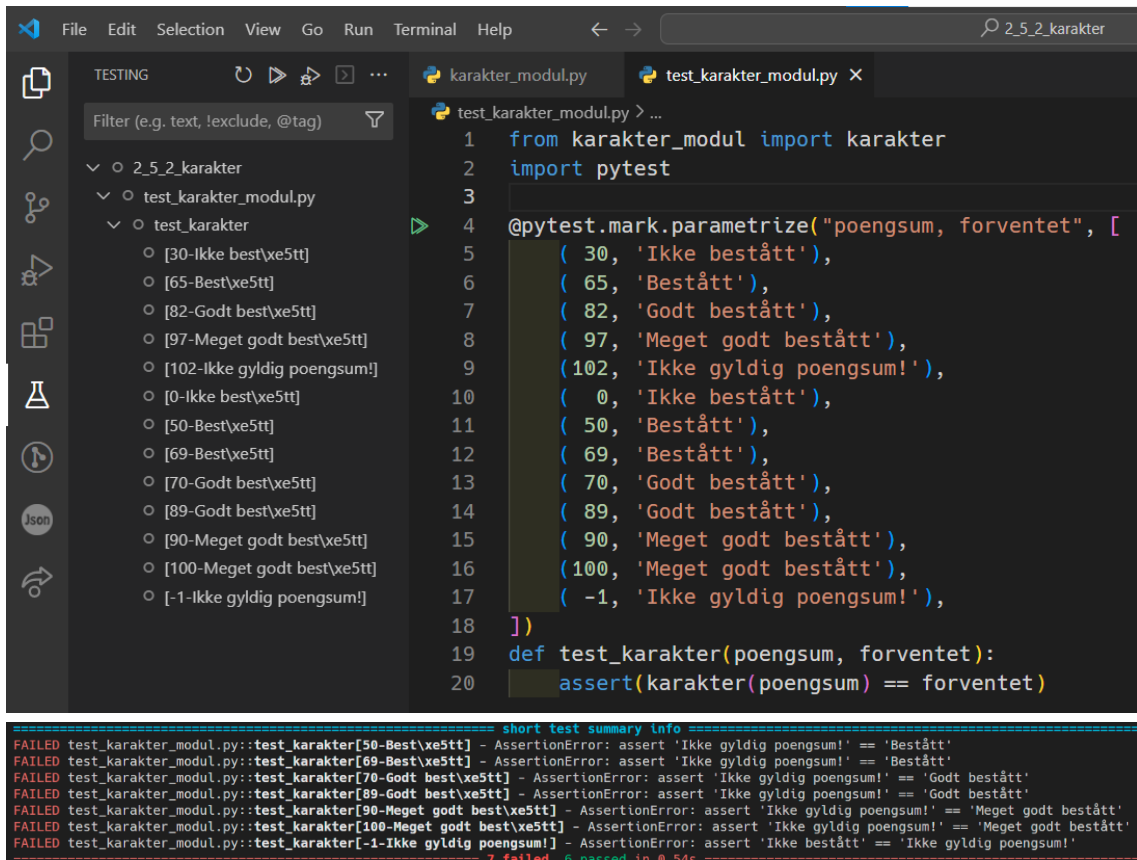
### b) Ved hjelp av `assert`-setninger

```
13 assert(karakter(30) == 'Ikke bestått')
14 assert(karakter(65) == 'Bestått')
15 assert(karakter(82) == 'Godt bestått')
16 assert(karakter(97) == 'Meget godt bestått')
17 assert(karakter(102) == 'Ikke gyldig poengsum!')
18 assert(karakter(0) == 'Ikke bestått')
19 #assert(karakter(50) == 'Bestått')
20 print(f'7: {karakter(50)}')
21 #assert(karakter(69) == 'Bestått')
22 print(f'8: {karakter(69)}')
23 # assert(karakter(70) == 'Godt bestått')
24 print(f'9: {karakter(70)}')
25 # assert(karakter(89) == 'Godt bestått')
26 print(f'10: {karakter(89)}')
27 # assert(karakter(90) == 'Meget godt bestått')
28 print(f'11: {karakter(90)}')
29 # assert(karakter(100) == 'Meget godt bestått')
30 print(f'12: {karakter(100)}')
31 # assert(karakter(-1) == 'Ikke gyldig poengsum!')
32 print(f'13: {karakter(-1)}')
```

```
▼ TERMINAL
7: Ikke gyldig poengsum!
8: Ikke gyldig poengsum!
9: Ikke gyldig poengsum!
10: Ikke gyldig poengsum!
11: Ikke gyldig poengsum!
12: Ikke gyldig poengsum!
13: Ikke bestått
```

Se vedlagte `assert_setninger.py` i mappen `2_5_2_karakter`

### c) Ved hjelp av `pytest` med parametere



The screenshot shows a VS Code editor with a file named `test_karakter_modul.py` open. The file contains a `pytest` parametrized test function `test_karakter` that tests the `karakter` function with various inputs and expected outputs. The test is parametrized with a list of tuples: `(poengsum, forventet)`. The test function is defined as `def test_karakter(poengsum, forventet): assert(karakter(poengsum) == forventet)`.

The left sidebar shows the file explorer with the following structure:

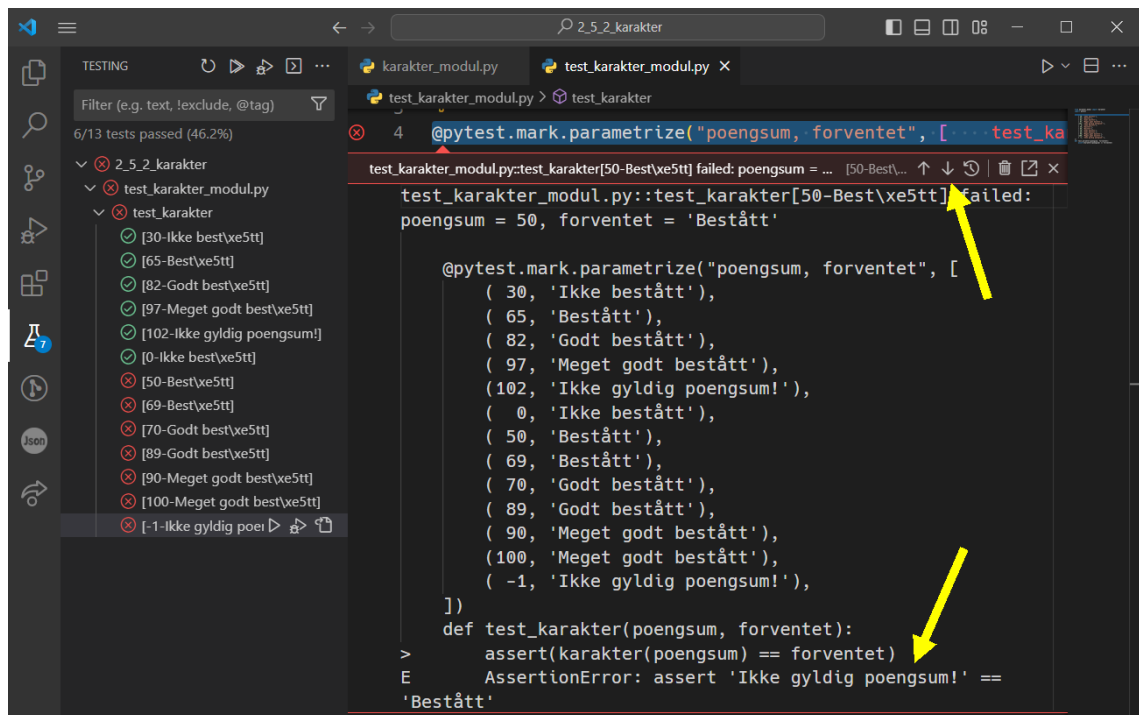
- 2\_5\_2\_karakter
  - test\_karakter\_modul.py
    - test\_karakter
      - [30-Ikke bestått]
      - [65-Bestått]
      - [82-Godt bestått]
      - [97-Meget godt bestått]
      - [102-Ikke gyldig poengsum!]
      - [0-Ikke bestått]
      - [50-Bestått]
      - [69-Bestått]
      - [70-Godt bestått]
      - [89-Godt bestått]
      - [90-Meget godt bestått]
      - [100-Meget godt bestått]
      - [-1-Ikke gyldig poengsum!]

The bottom of the screenshot shows the output of the test run, indicating that 7 tests failed and 6 passed. The failed tests are:

- `test_karakter_modul.py::test_karakter[50-Bestått]` - AssertionError: assert 'Ikke gyldig poengsum!' == 'Bestått'
- `test_karakter_modul.py::test_karakter[69-Bestått]` - AssertionError: assert 'Ikke gyldig poengsum!' == 'Bestått'
- `test_karakter_modul.py::test_karakter[70-Godt bestått]` - AssertionError: assert 'Ikke gyldig poengsum!' == 'Godt bestått'
- `test_karakter_modul.py::test_karakter[89-Godt bestått]` - AssertionError: assert 'Ikke gyldig poengsum!' == 'Godt bestått'
- `test_karakter_modul.py::test_karakter[90-Meget godt bestått]` - AssertionError: assert 'Ikke gyldig poengsum!' == 'Meget godt bestått'
- `test_karakter_modul.py::test_karakter[100-Meget godt bestått]` - AssertionError: assert 'Ikke gyldig poengsum!' == 'Meget godt bestått'

The output also shows the total result: `7 failed, 6 passed in 0.54s`.

## Smidig IT-2



Se vedlagte test\_karakter\_modu.py i mappen 2\_5\_2\_karakter