Rolf Vidar Hoksaas
*Institutt for informatikk*
*Universitetet i Oslo*

**1. September 2019**

## OBLIGER 2

Activity 3 — Kodeflyt

```
    1   def print_prosa():
4, 10       print("Melding til alle gaardeiere:")
5, 11       print("Antall dyr paa gaarden: ")
    #
    2   antall_dyr = 4
    3   print_prosa()
    6   print(antall_dyr)
    7   antall_nye_dyr = int(input("Hvor mange nye dyr kommer til gaarden: "))
    8   antall_dyr = antall_dyr + antall_nye_dyr
    9   print_prosa()
   10   print(antall_dyr)
    #
   11   if antall_dyr > 12:
    #       print("Det er mer enn ett dusin dyr paa gaarden!")
   12   elif antall_dyr == 12:
   13       print("Det er ett dusin dyr paa gaarden!")
    #   else:
    #       print("Det er mindre enn ett dusin dyr paa gaarden!")
```

**Explanation** As we can see, the first thing that happens is that we create a function `print_prosa` which takes no arguments. This function prints prints two different strings separately when called. After this, we create an `Integer` variable with the value of `4` on the global scope. Next, we call the previously defined function `print_prosa`, and thus print the two given strings. On yet another line, we print the `4` stored in the variable `antall_dyr` (4) casted as a `String`.

```
Melding til alle gaardeiere:
Antall dyr paa gaarden:
4
Hvor mange nye dyr kommer til gaarden :␣
```

As the user inputs `8` and hits ↵, the number will be returned from the input function as a string, and this value will then be casted to an `Integer` (since `"8"` is int-like), and then stored into a new variable named `antall_nye_dyr`. Following up, this newly created variable will be added together to the previously defined variable, `antall_dyr`. Since $4 + 8 \implies 12$, the `Integer` 12 is stored into `antall_dyr`.

Again, we call the function `print_prosa` which prints two different strings separately, and on yet another line we pipe the value of `antall_dyr` to the terminal.

```
Melding til alle gaardeiere:
Antall dyr paa gaarden:
4
Hvor mange nye dyr kommer til gaarden : 8↵
Melding til alle gaardeiere:
Antall dyr paa gaarden:
12
```

After this, reaching line 11, the code checks whether the value stored in `antall_dyr` is greater than 12, and since $12 > 12 \implies \mathit{False}$, it checks the condition of the corresponding condition of the statement, namely checking if `antall_dyr` equals `12`. Since this is True, it runs the indented block of code, and thus prints `"Det er ett dusin dyr paa gaarden!"` on screen.

Thanks to lazy evaluation, the last else statement is ignored.

```
Melding til alle gaardeiere:
Antall dyr paa gaarden:
4
Hvor mange nye dyr kommer til gaarden : 8
Melding til alle gaardeiere:
Antall dyr paa gaarden:
12
Det er ett dusin dyr paa gaarden!
```

□

**Suggestions** I have some comments regarding this piece of code, and I would like to direct them by providing a much better solution for the task at hand.

```
MORE_THAN_12 = "Det er mer enn ett dusin dyr paa gaarden!"
EXACTLY_12 = "Det er ett dusin dyr paa gaarden!"
LESS_THAN_12 = "Det er mindre enn ett dusin dyr paa gaarden!"

def print_prosa(n_dyr = None):
      print("Melding til alle gaardeiere:"\
      f"Antall dyr paa gaarden: {n_dyr}")

antall_dyr = 4
print_prosa(antall_dyr)

antall_dyr += int(input("Hvor mange nye dyr kommer til gaarden: "))
print_prosa(antall_dyr)

if antall_dyr > 12:
    print(MORE_THAN_12)
elif antall_dyr == 12:
    print(EXACTLY_TWELVE)
else:
    print(LESS_THAN_TWELVE)
```

By using `'+='` instead of storing a new variable to then add to the old variable, we use less memory, are less prone to doing spelling errors, and the code is much more readable. Especially considering we were never going to use the old variable again.

By doing a single print statement, instead of three, we have fewer locations something can go wrong. Furthermore, since the suggested prosa implicitly asks for a trailing number it makes no sense to not print this value in the same function. If no value is passed to the function, use a default value `0`.

Lastly, moving static strings to the top of the document facilitates changing these on-demand, such as during translation of the program, and also enforces integrity throughout the program if the string were to be reused at a later stage of the program.

These changes add readability to the code, and also facilitates maintenance of the code. □

*Submitted by Rolf Vidar Hoksaas on 1. September 2019.*