En rolig start

OPPSTART 16:30 :)

Introduksjon til programmering

KRISTIAN BOTNEN, 2024

NITO

60 minutter

3 moduler

Passer for alle

Dybdekurs

Om veiviseren

I denne veiviseren vil du lære om hva kunstig intelligens er, hvordan det fungerer – og hvordan kunstig intelligente systemer kan brukes på en etisk og ansvarlig måte på arbeidsplassen.

https://digitalnorway.com/kurs/kunstig-intelligens-pa-arbeidsplassen-etikk-ansvar-og-verdiskaping/

PROGRAMMERING I GRUNNSKOLEN - NY LÆREPLAN

	Matematikk	Naturfag	Musikk	Kunst og håndverk
2	lage og følgje reglar og trinnvise instruksjonar i leik og spel			
3	lage og følgje reglar og trinnvise instruksjonar i leik og spel knytte til koordinatsystemet		Eksperimentere med rytmer, melodier og andre grunnelementer, sette sammen mønstre til komposisjoner, også ved bruk av digitale verktøy, og beskrive arbeidsprosesser og resultater	
4	lage algoritmar og uttrykkje dei ved bruk av variablar, vilkår og lykkjer			
5	lage og programmere algoritmar med bruk av variablar, vilkår og lykkjer	Utforske, lage og programmere teknologiske systemer som består av deler som virker sammen	Bruke teknologi og digitale verktøy til å skape, øve inn og bearbeide musikk	Bruke programmering til å skape interaktivitet og visuelle uttrykk
6	bruke variablar, lykkjer, vilkår og funksjonar i programmering til å utforske geometriske figurar og mønster			
7	bruke programmering til å utforske data i tabellar og datasett			
8	utforske korleis algoritmar kan skapast, testast og forbetrast ved hjelp av programmering	Utforske, forstå og lage teknologiske systemer som består av en sender og en mottaker Bruke programmering til å utforske naturfaglige fenomener	Skape og programmere musikalske forløp ved å eksperimentere med lyd fra ulike kilder	Utforske hvordan digitale verktøy og ny teknologi kan gi muligheter for kommunikasjonsformer og opplevelser i skapende prosesser og produkter
9	simulere utfall i tilfeldige forsøk og berekne sannsynet for at noko skal inntreffe ved å bruke programmering			
10	utforske matematiske eigenskapar og samanhengar ved å bruke programmering			

Hvem er vi

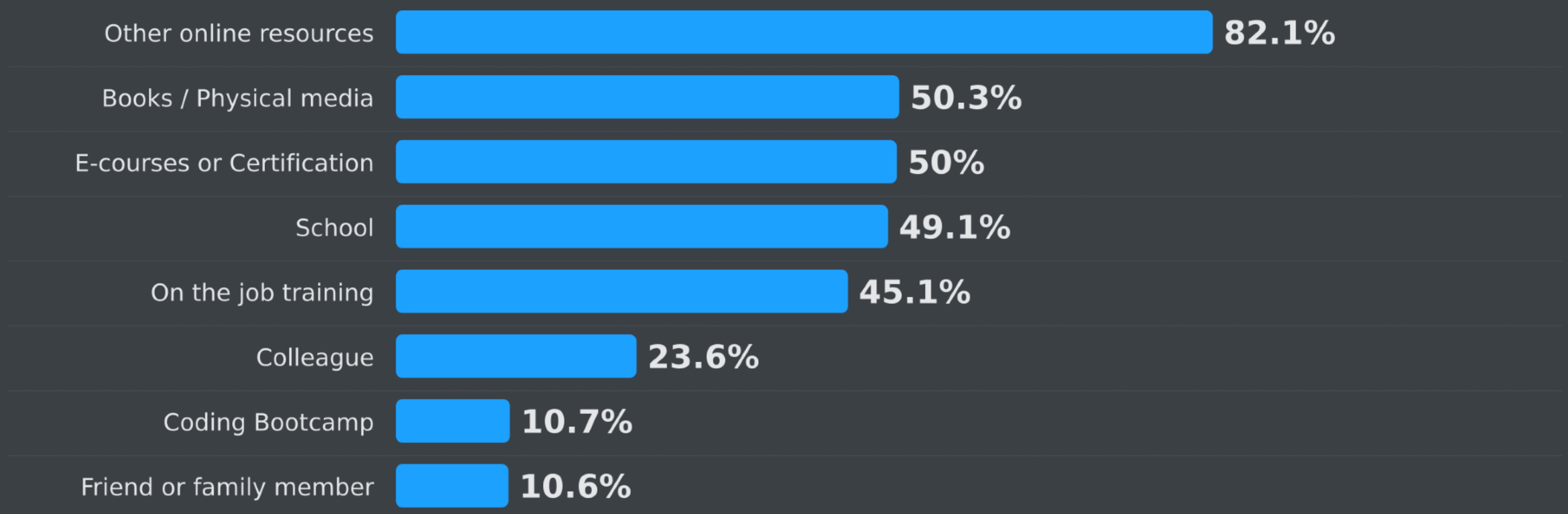
Hvem er "dere"?

https://survey.stackoverflow.co/2024/

Hvem er "dere"?

Learning to code / All Respondents

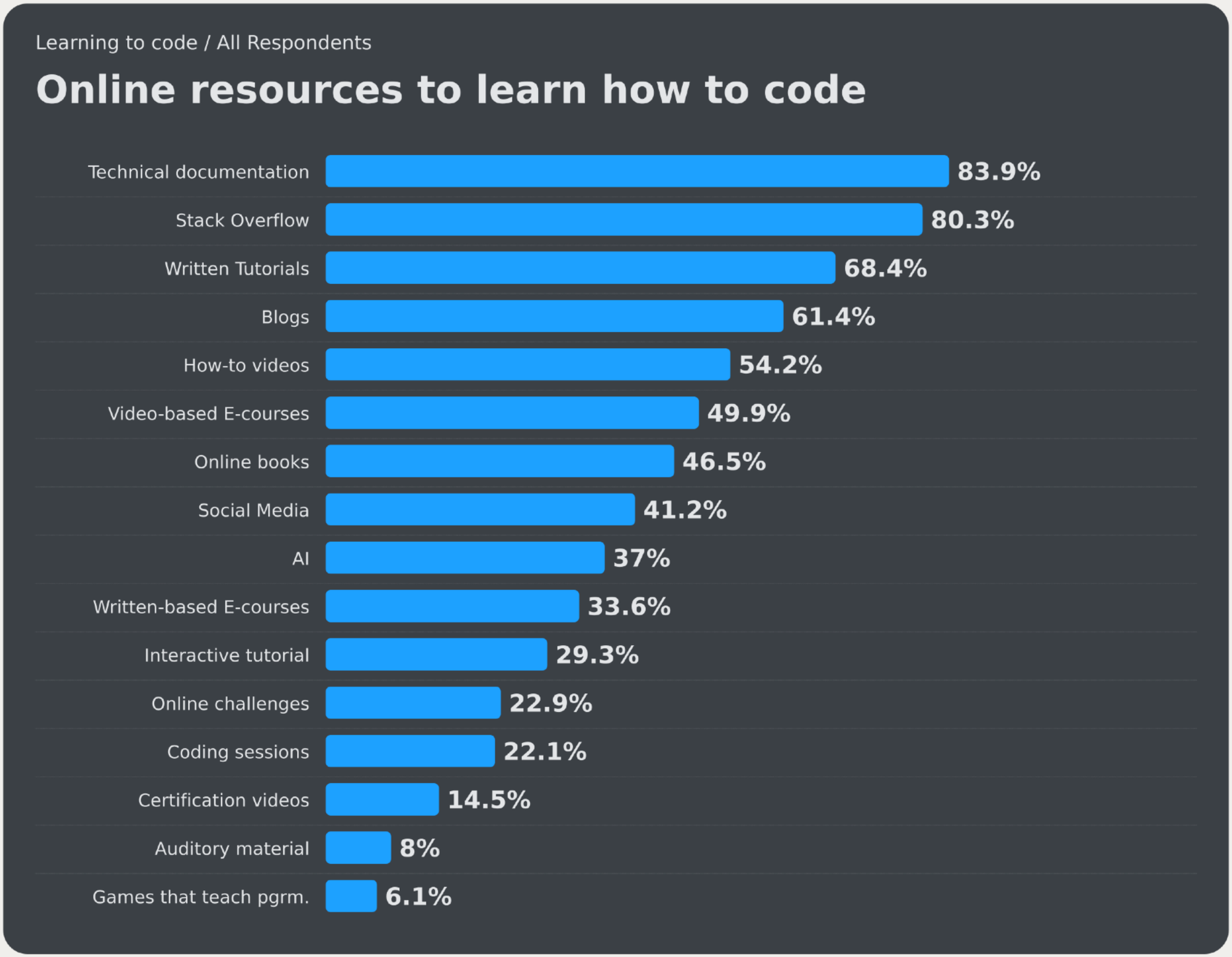
Learning to code



2024 Developer Survey

Source: survey.stackoverflow.co/2024
Data licensed under Open Database License (ODbL)

Hvem er "dere"?


Learning to code / All Respondents
Online resources to learn how to code
Technical documentation 83.9%
Stack Overflow 80.3%
Written Tutorials 68.4%
Blogs 61.4%
How-to videos 54.2%
Video-based E-courses 49.9%
Online books 46.5%
Social Media 41.2%
AI 37%
Written-based E-courses 33.6%
Interactive tutorial 29.3%
Online challenges 22.9%
Coding sessions 22.1%
Certification videos 14.5%
Auditory material 8%
Games that teach pgrm. 6.1%
2024 Developer Survey
Source: survey.stackoverflow.co/2024
Data licensed under Open Database License (ODbL)

Hvem er "dere"?

AI tools in the development process

76% of all respondents are using or are planning to use AI tools in their development process this year, an increase from last year (70%). Many more developers are currently using AI tools this year, too (62% vs. 44%).

Do you currently use AI tools in your development process? *

Benefits of AI tools

81% agree increasing productivity is the biggest benefit that developers identify for AI tools. Speeding up learning is seen as a bigger benefit to developers learning to code (71%) compared to professional developers (61%).

For the AI tools you use as part of your development workflow, what are the MOST important benefits you are hoping to achieve? Please check all that apply.

Accuracy of AI tools

Similar to last year, developers remain split on whether they trust AI output: 43% feel good about AI accuracy and 31% are skeptical. Developers learning to code are trusting AI accuracy more than their professional counterparts (49% vs. 42%).

How much do you trust the accuracy of the output from AI tools as part of your development workflow?

Are AI tools a threat to your job

70% of professional developers do not perceive AI as a threat to their job.

Do you believe AI is a threat to your current job?

https://survey.stackoverflow.co/2024/ai/

Hvem er dere?

https://www.menti.com/ala8ng79hftk

Dagens agenda

- Hva er python
- Noen kule datatyper
- Variabler + oppgave 1
- Kodeblokker & løkker + oppgave 2
- If - then - else + oppgave + oppgave 3
- Funksjoner og moduler + bonusoppgave
- Oppsummering og veien videre

Kursmateriell

https://github.com/kbotnen/pythonkveldskurs

Hva er Python

Python elevator pitch

Easy Efficient Interpreted Readable

Free as in beer Community

Extensible Cross-plattform Portable

Well documented Open Source

KOMPILERT KODE

- Raskere kode
- Klar til å kjøres når brukeren får programmet
- Et ekstra steg i utviklingen
- Låst til plattform
- Kildekoden er lukket
- Eks: C++, Java, Swift

TOLKET KODE

- Enklere å utvikle, enklere å teste
- Ikke låst til plattform
- Brukeren trenger en tolker for å kjøre programmet
- Kildekoden er åpen
- Eks: Ruby, Matlab, Python

```
(base) kristianbotnen@c02vl0lphtdg ~ % python
Python 3.9.12 (main, Jun  1 2022, 06:36:29)
[Clang 12.0.0 ] :: Anaconda, Inc. on darwin
Type "help", "copyright", "credits" or "license" for more information.
```

```
>>> import this
The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!
>>>
```

```
>>> quit()
(base) kristianbotnen@c02vl0lphtdg ~ %
```

PEP 8 – Style Guide for Python Code

Author: Guido van Rossum <guido at python.org>, Barry Warsaw <barry at python.org>, Nick Coghlan <ncoghlan at gmail.com>

Status: Active
Type: Process
Created: 05-Jul-2001
Post-History: 05-Jul-2001, 01-Aug-2013

▶ **Table of Contents**

Introduction

This document gives coding conventions for the Python code comprising the standard library in the main Python distribution. Please see the companion informational PEP describing style guidelines for the C code in the C implementation of Python.

This document and PEP 257 (Docstring Conventions) were adapted from Guido's original Python Style Guide essay, with some additions from Barry's style guide [2].

This style guide evolves over time as additional conventions are identified and past conventions are rendered obsolete by changes in the language itself.

Many projects have their own coding style guidelines. In the event of any conflicts, such project-specific guides take precedence for that project.

A Foolish Consistency is the Hobgoblin of Little Minds

One of Guido's key insights is that code is read much more often than it is written. The guidelines provided here are intended to improve the readability of code and make it consistent across the wide spectrum of Python code. As PEP 20 says, "Readability counts".

A style guide is about consistency. Consistency with this style guide is important. Consistency within a project is more important. Consistency within one module or function is the most important.

However, know when to be inconsistent – sometimes style guide recommendations just aren't applicable. When in doubt, use your best judgment. Look at other examples and decide what looks best. And don't hesitate to ask!

In particular: do not break backwards compatibility just to comply with this PEP!

https://docs.conda.io/en/latest/miniconda.html

https://www.jetbrains.com/pycharm/

https://code.visualstudio.com/

https://replit.com/

https://codewith.mu/en/

Noen kule typer

1 = Tall
1.0 = Tall med desimal
a = Bokstav
hei = Ord
hei du = Setning

VERDIER

1 = int
1.0 = float
a = string
hei = string
hei du = string

TYPER

Typer

```
:::python
Text Type:          str
Numeric Types:      int, float, complex
Sequence Types:     list, tuple, range
Mapping Type:       dict
Set Types:          set, frozenset
Boolean Type:       bool
Binary Types:       bytes, bytearray, memoryview
None Type:          NoneType
```

Variabler

Variabelnavn
1

Variabelnavn
a

Variabelnavn
Kristian

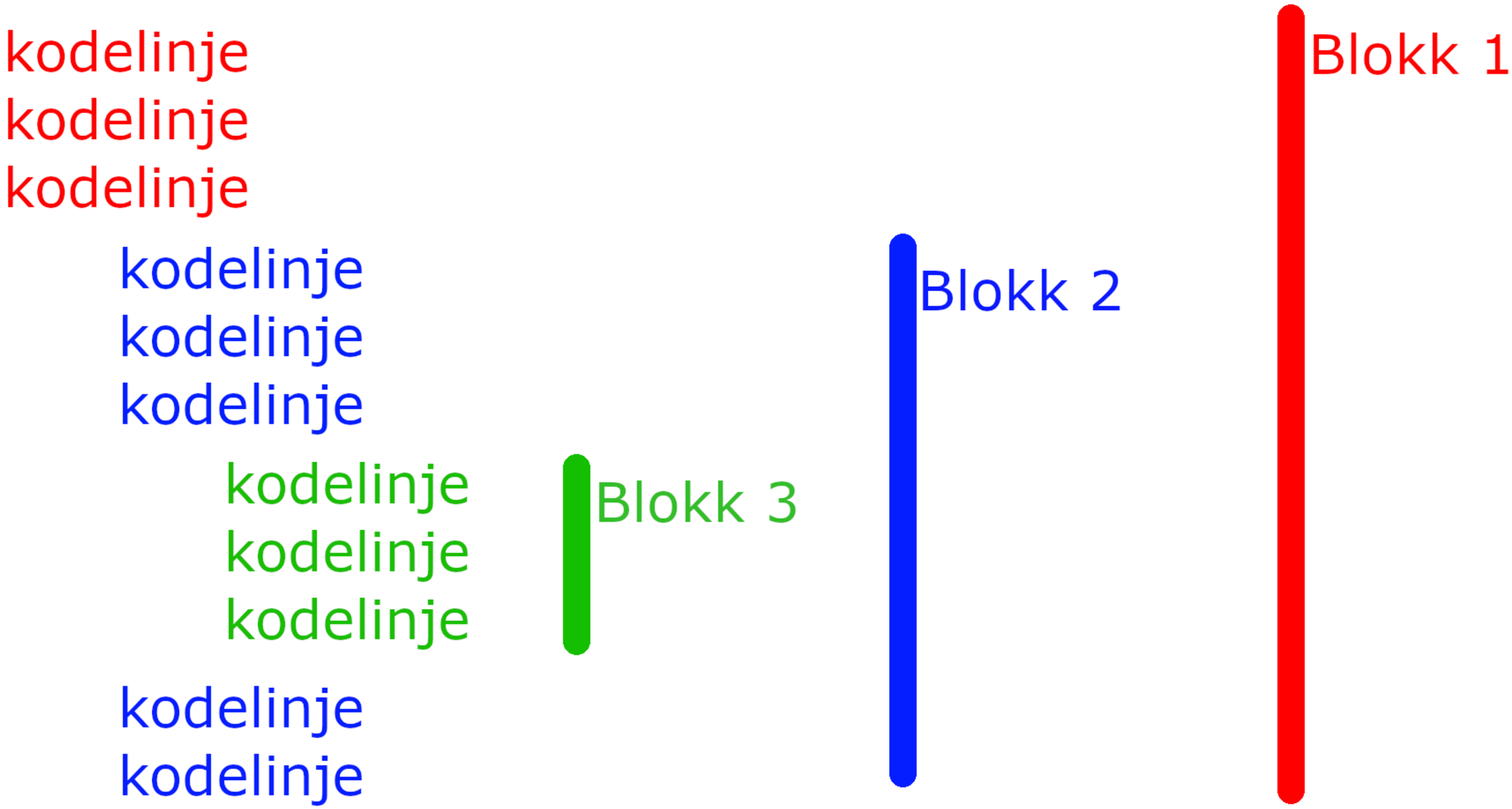
variabel_a	variabel_b	variabel_c
1	"a"	"Kristian"

```
# Pythonkode
variabelnavn_a = 1
variabelnavn_b = "a"
variabelnavn_c = "Kristian"
```

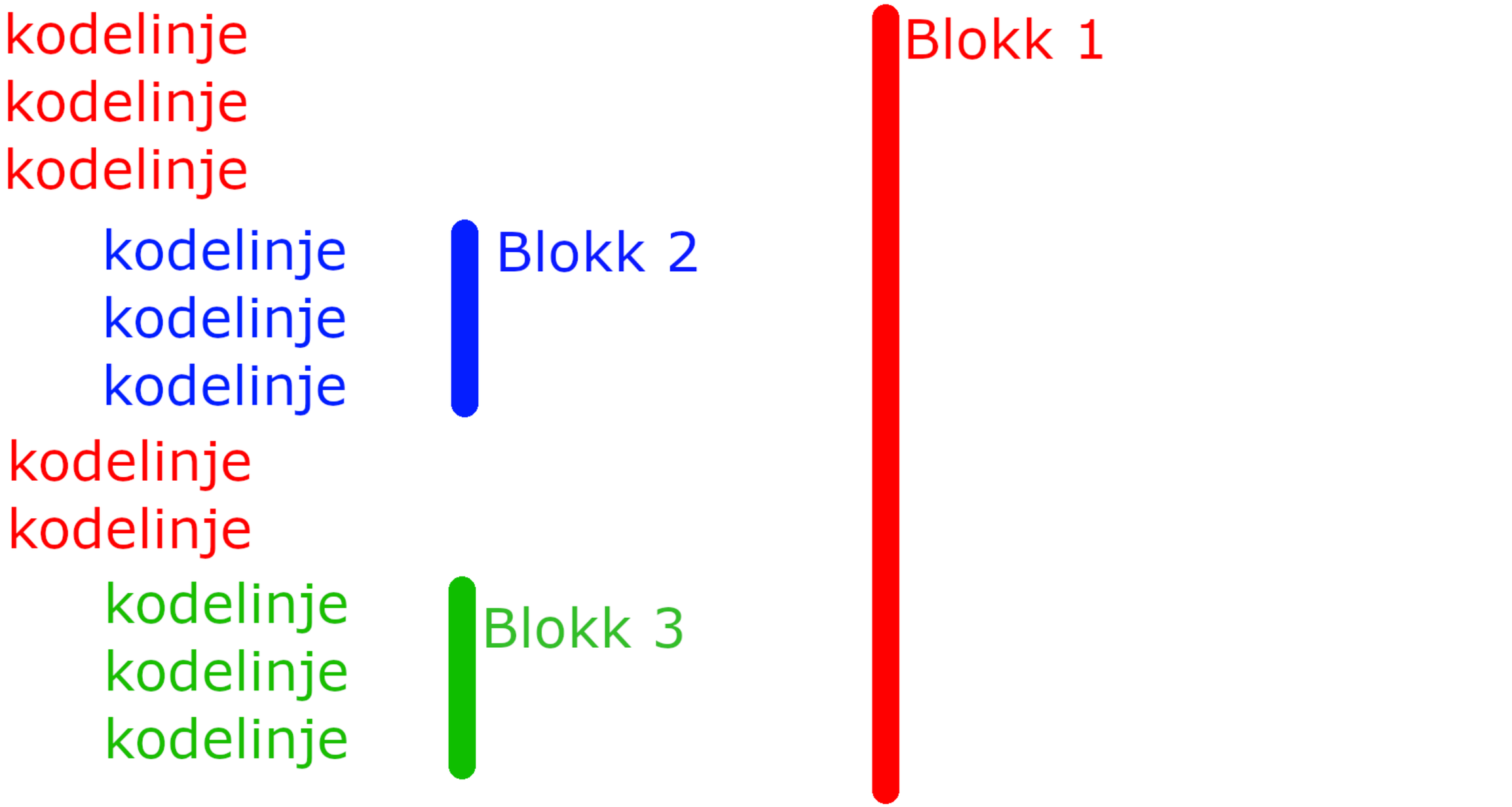
```
# Pythonkode

variabelnavn_a = 1
variabelnavn_b = "a"
variabelnavn_c = "Kristian"

variabelnavn_a = "Python er gøy"
```

Kodeblokker & løkker

Kodeblokk


kodelinje
kodelinje
kodelinje
kodelinje
kodelinje
kodelinje
kodelinje
kodelinje
kodelinje
kodelinje
kodelinje
kodelinje
Blokk 1
Blokk 2
Blokk 3

Kodeblokk


kodelinje
kodelinje
kodelinje
kodelinje
kodelinje
kodelinje
Blokk 2
kodelinje
kodelinje
kodelinje
kodelinje
kodelinje
kodelinje
Blokk 3
Blokk 1

Tilstandsløkke

Ta på lue så lenge det snør ute.

Så lenge det snør ute så ta på lue

Om det ikke snør ute, ikke ta på lue

Tilstandsløkke

```
Linje 1        i = 0
Linje 2        while (i < 5):
Linje 3            print(i)
Linje 4            i = i + 1
```

Telleløkke

```
for i in range(0, 5):
    print(i)
```

If - then - else

If - then - else

```
alder = 18
if alder > 18:
    print("Du er myndig")
else:
    print("Du er ikke myndig")
```

If - then - else

Operator	Navn	Eksempel
==	Equal	x == y
!=	Not Equal	x != y
>	Greater than	x > y
<	Less than	x < y
>=	Greater than or equal to	x >= y
<=	Less than or equal to	X <= y

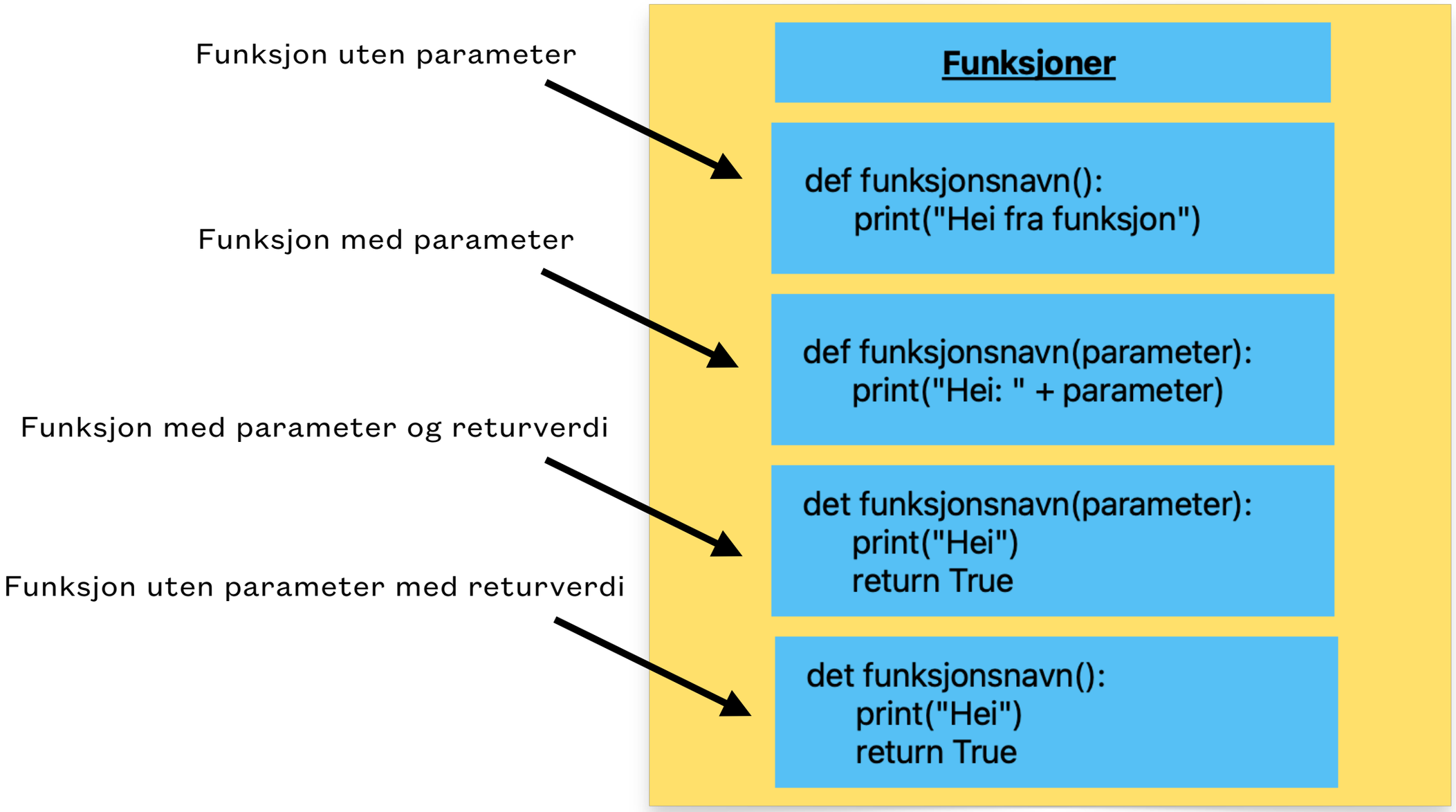
Funksjoner og moduler

Funksjoner

```python
def greet_function():
    print("Hello from a function")

greet_function()
```

```python
def fibonacci(n):
    if n <= 1: # If the number is 0, then the answer is 0. If the number is 1, then the answer is 1.
        return n
    else:
        return fibonacci(n - 1) + fibonacci(n - 2) # Each successive fibonacci number is found by adding up the two numbers before it.

print('Fibonacci sequence:')
for i in range(5):
    print(fibonacci(i))
```

Funksjoner

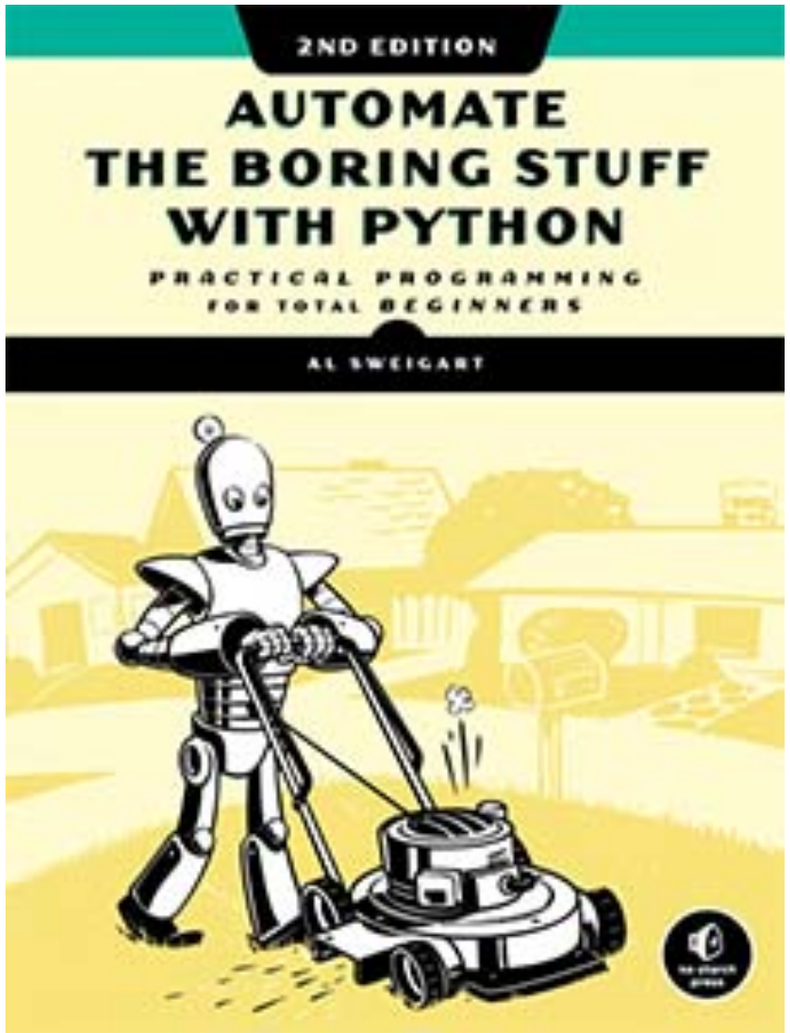

Funksjon uten parameter
Funksjon med parameter
Funksjon med parameter og returverdi
Funksjon uten parameter med returverdi
Funksjoner
def funksjonsnavn():
print("Hei fra funksjon")
def funksjonsnavn(parameter):
print("Hei: " + parameter)
det funksjonsnavn(parameter):
print("Hei")
return True
det funksjonsnavn():
print("Hei")
return True

```
import random

for i in range(10):
    print(random.randint(1, 25))
```

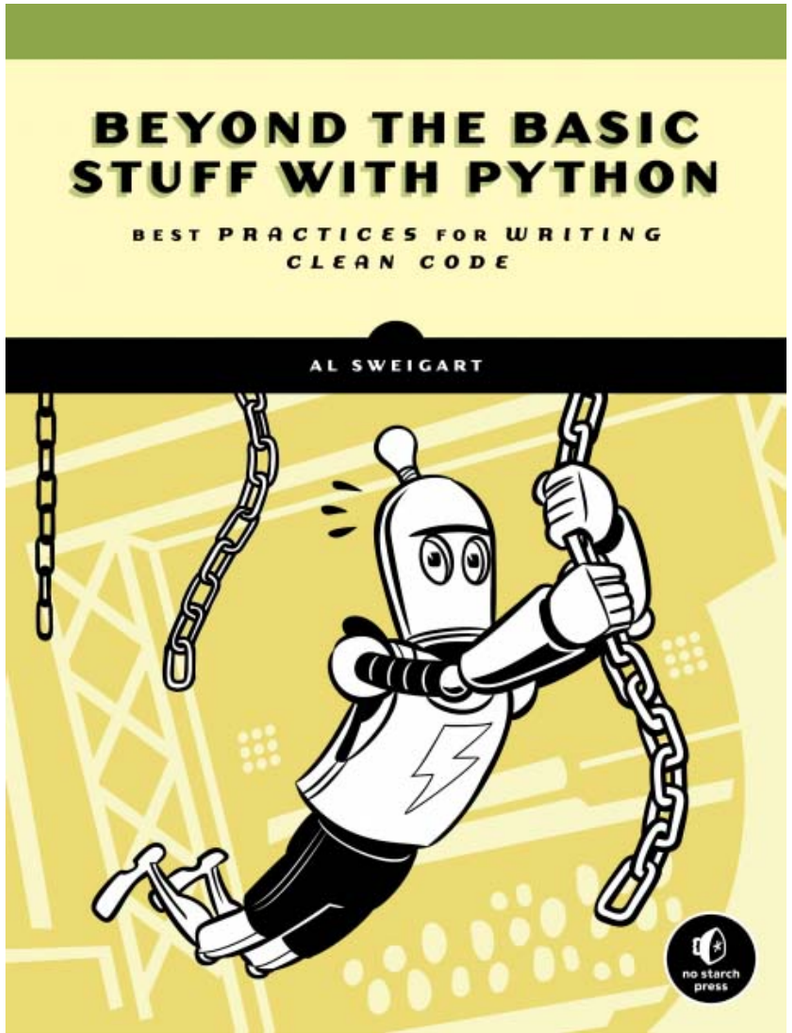
Moduler

```
import numpy as np

x = np.array([1, 2, 3])
print(x)
```

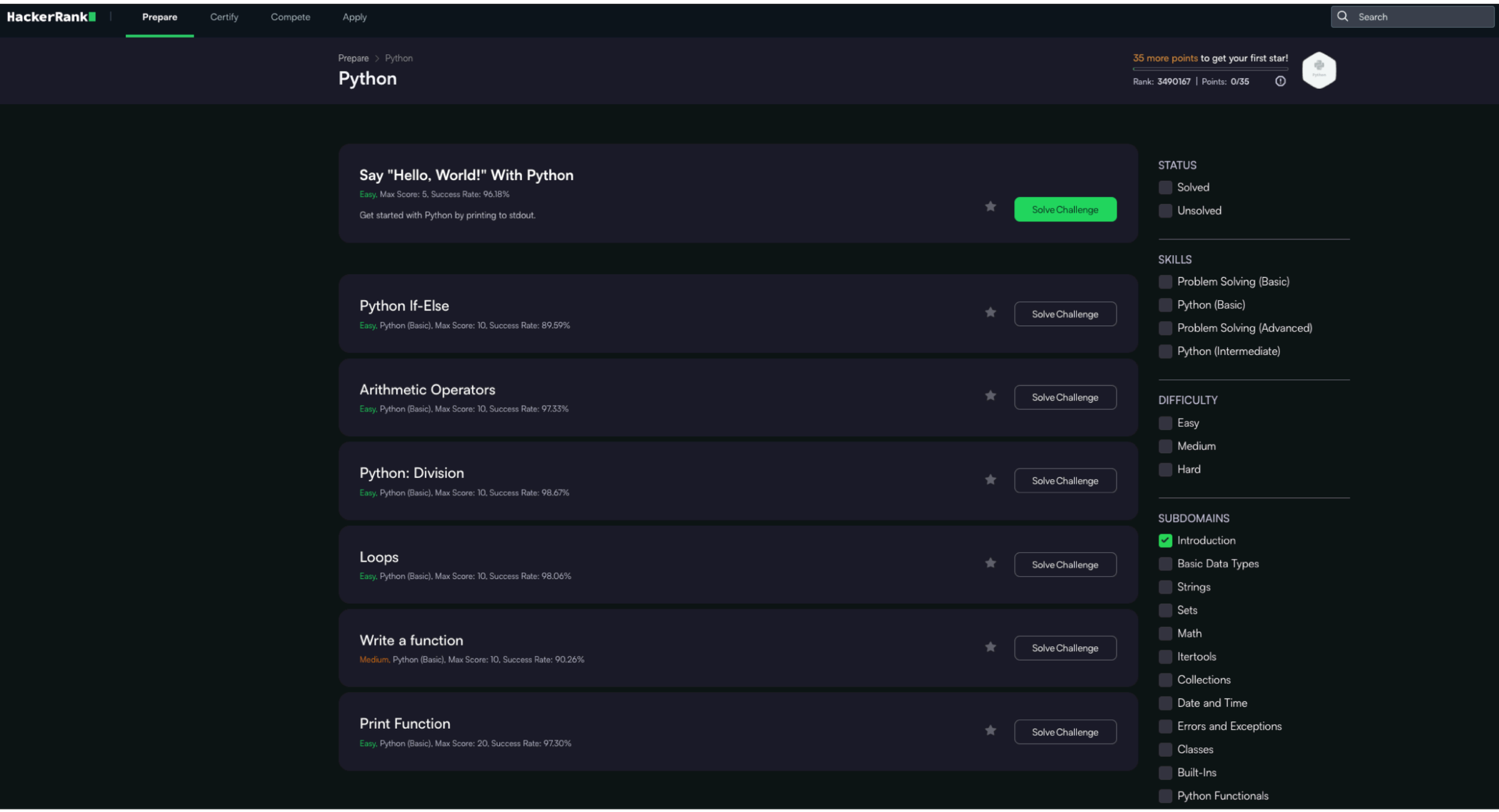
Veien videre



https://automatetheboringstuff.com



https://inventwithpython.com/beyond/



https://www.hackerrank.com/

--- Day 1: Trebuchet?! ---

Something is wrong with global snow production, and you've been selected to take a look. The Elves have even given you a map; on it, they've used stars to mark the top fifty locations that are likely to be having problems.

You've been doing this long enough to know that to restore snow operations, you need to check all fifty stars by December 25th.

Collect stars by solving puzzles. Two puzzles will be made available on each day in the Advent calendar; the second puzzle is unlocked when you complete the first. Each puzzle grants one star. Good luck!

You try to ask why they can't just use a weather machine ("not powerful enough") and where they're even sending you ("the sky") and why your map looks mostly blank ("you sure ask a lot of questions") and hang on did you just say the sky ("of course, where do you think snow comes from") when you realize that the Elves are already loading you into a trebuchet ("please hold still, we need to strap you in").

As they're making the final adjustments, they discover that their calibration document (your puzzle input) has been amended by a very young Elf who was apparently just excited to show off her art skills. Consequently, the Elves are having trouble reading the values on the document.

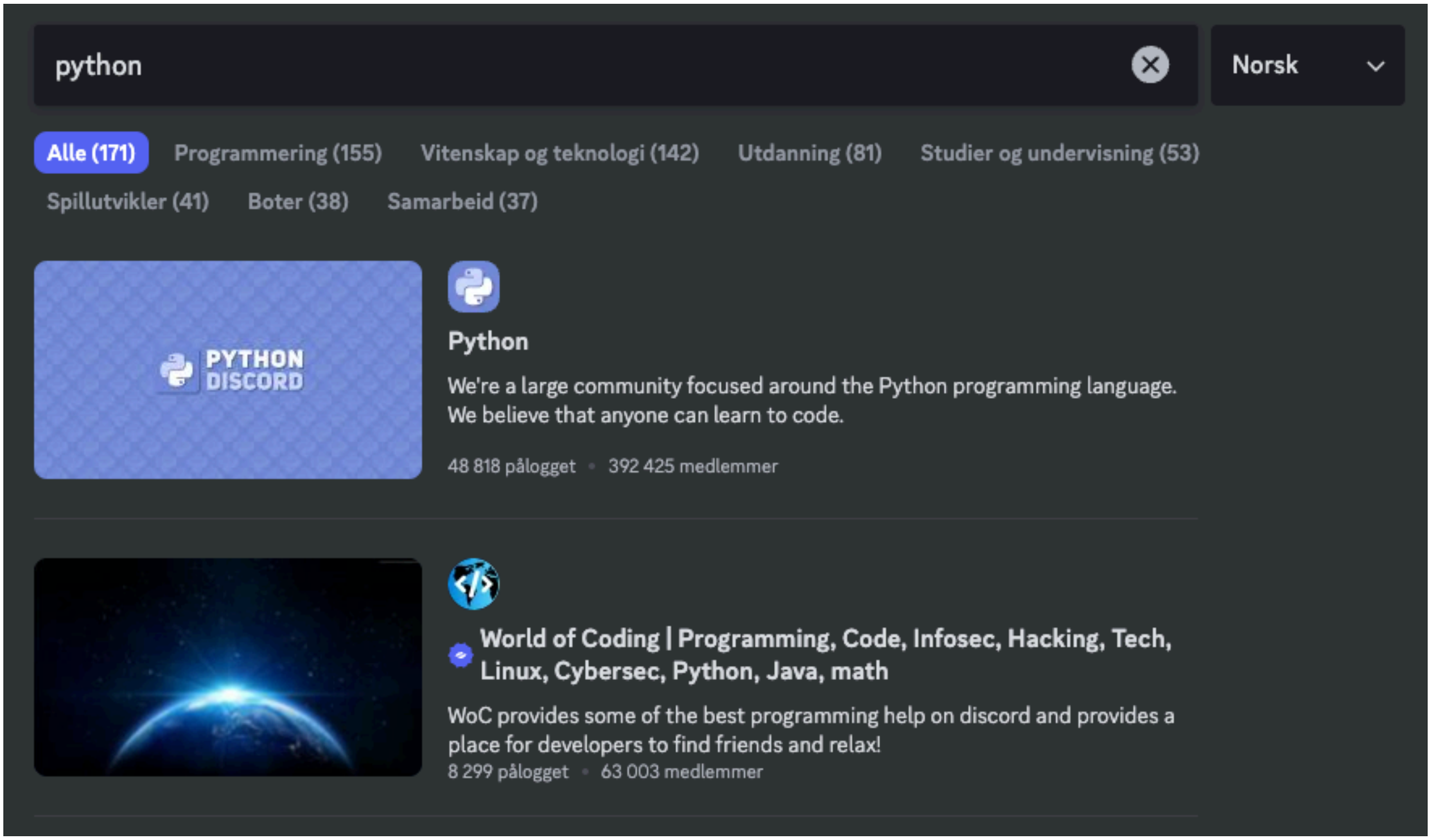
The newly-improved calibration document consists of lines of text; each line originally contained a specific calibration value that the Elves now need to recover. On each line, the calibration value can be found by combining the first digit and the last digit (in that order) to form a single two-digit number.

For example:

```
1abc2
pqr3stu8vwx
a1b2c3d4e5f
treb7uchet
```

In this example, the calibration values of these four lines are `12`, `38`, `15`, and `77`. Adding these together produces `142`.

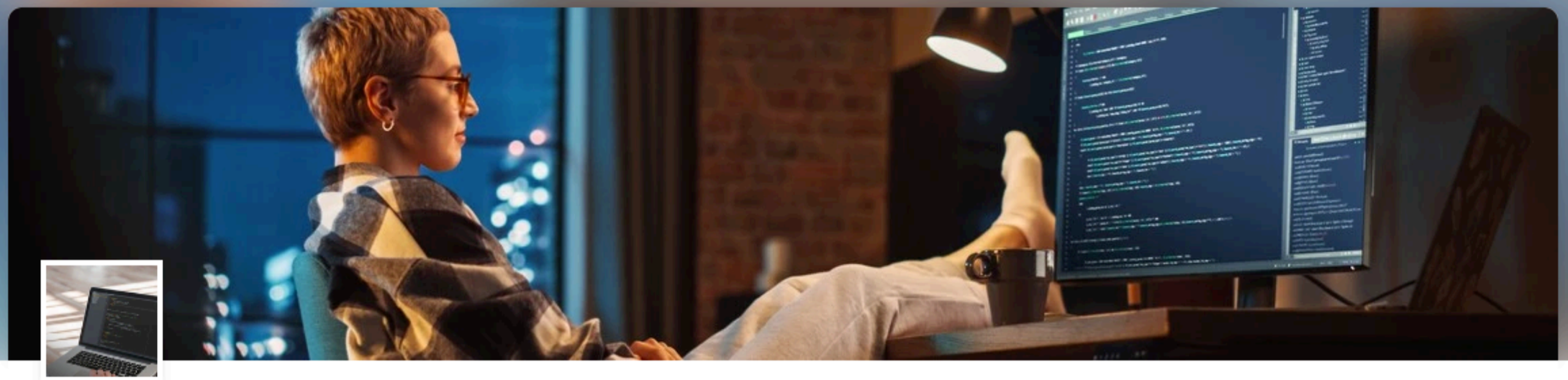
Consider your entire calibration document. What is the sum of all of the calibration values?



https://support.discord.com/

NITO

Hovedside Livefeed Medlemsoversikt Arrangementer Mediesenter Fagnettverk Nyheter



NITO Python

+135 Har blitt med

Livefeed Medlemmer Arrangementer Mediearkiv Forum

Skriv et innlegg

Om

NITO Python er et spisset fagnettverk for deg som er interessert i Python-koding. Denne gruppen i NITO Puls/Hivebrite ble opprettet ifm "Introduksjon til Python" gratiskurset som ble avholdt 6. mars 2024.

Oktober

02 OKT

Introduksjon til Python

2. oktober 2024, Norge

Virtuelt

14 OKT

Slik lager du webapplikasjoner med Python

14. oktober 2024, Norge

Webinar

November

05 NOV

Python for ingeniøren

5. november 2024, Oslo

Oslo sentrum

Desember

02 DES

Kunstig intelligens og maskinlæring med Python

2. desember 2024, Norge

Virtuelt

https://www.nito.no/kurs-og-arrangementer/

- https://github.com/kbotnen/pythonkveldskurs

- https://oppgaver.kidsakoder.no/python
- https://www.w3schools.com/python/default.asp

- https://www.learnpython.org
- https://replit.com/learn/100-days-of-python

- https://www.reddit.com/r/Python/
- https://www.pythondiscord.com/resources/