

En rolig start

---

# Introduksjon til programmering II

---

KRISTIAN BOTNEN, 2025

---

# Øktens agenda

---

**Noen flere datatyper**

---

**Dokumentasjon og hjelp**

---

**Veien videre**

---



[https://github.com/kbotnen/python\\_geovitenskap](https://github.com/kbotnen/python_geovitenskap)

---

---

# Noen flere datatyper

---

---

# Operatorer

`:::python`

Arithmetic operators `(+, -, *, /, %, **, //)`

Assignment operators `(=, +=, -=, *=, /=, %=, //=, **=, &=, |=, ^=, >>=, <<=)`

Comparison operators `(==, !=, >, <, >=, <=)`

Logical operators `(and, or, not)`

Identity operators `(is, is not)`

Membership operators `(in, not in)`

Bitwise operators `(&, |, ^, ~, <<, >>)`

---

# Python Operators and Booleans Cheat Sheet by [Nouha Thabet](#)

## Python Arithmetic Operators

Addition	9 + 2	>> 11
Subtraction	9 - 2	>> 7
Multiplication	9 * 2	>> 18
Division	9 / 2	>> 4.5
Modulus	9 % 2	>> 1
Exponentiation	3 ** 2	>> 81
Floor division	9 // 2	>> 4

## Python Assignment Operators

Operator	Example	Same As
=	x = 2	x = 2
+=	x += 2	x = x + 2
-=	x -= 2	x = x - 2
*=	x *= 2	x = x * 2
/=	x /= 2	x = x / 2
%=	x %= 2	x = x % 2
//=	x //= 2	x = x // 2
**=	x **= 2	x = x ** 2

## Python Comparison Operators

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

## Boolean Values

In programming you often need to know if an expression is `True` OR `False`.

You can evaluate any expression in Python, and get the answer.

```
print(5 < 8)          >>> True
print(5 > 8)          >>> False
```

## Python Logical Operators

**and** Returns True if both statements are true

```
x < 5 and x < 10
```

**or** Returns True if one of the statements is true

```
x < 5 or x < 4
```

**not** Reverse the result, returns False if the result is true

```
not(x < 5 and x < 10)
```

## Python Identity Operators

**is** Returns true if both variables are the same object

```
x is y
```

**is not** Returns true if both variables are not the same object

```
x is not y
```

## Python Membership Operators

**in** Returns True if a sequence with the specified value is present in the object

```
x in y
```

**not in** Returns True if a sequence with the specified value is not present in the object

```
x not in y
```

## Python Bitwise Operators

&	AND	Sets each bit to 1 if both bits are 1
	OR	Sets each bit to 1 if one of two bits is 1
^	XOR	Sets each bit to 1 if only one of two bits is 1
~	NOT	Inverts all the bits
<<	Zero fill left shift	Shift left by pushing zeros in from the right and let the leftmost bits fall off
>>	Signed right shift	Shift right by pushing copies of the leftmost bit in from the left, and let the rightmost bits fall off

---

# Int / Float / Complex

```
:::python
var_int = 1
var_float = 1.0
var_complex = 1j

print(var_int, var_float, var_complex, sep=', ')
print(type(var_int), type(var_float), type(var_complex), sep=', ')
```

---

# String

```
:::python
print("Hello")
print('Hello')
print("""Lorem ipsum dolor sit amet,
consectetur adipiscing elit,
sed do eiusmod tempor incididunt
ut labore et dolore magna aliqua.""")
```



---

# Boolean

```
#!/usr/bin/env python
var_string = "Hello World!"
if ('ello' in var_string): # The values between ( and ) will evaluate to True or False, in this example True
    print("We found 'ello' in our string")
else:
    print("We did not find 'ello' in our string")
```

---

<i>Index</i>	<i>Verdi</i>	<b>DUPLICATES</b>
0	«Henry»	<b>CHANGEABLE</b>
1	«Ford»	<b>ORDERED</b>
2	«English»	

List

<i>Nøkkel</i>	<i>Verdi</i>	<b>NO-DUPLICATES</b>
«f_Name»	«Henry»	<b>CHANGEABLE</b>
«l_Name»	«Ford»	<b>ORDERED</b>
«language»	«English»	

Dictionary

# Collections

<i>Index</i>	<i>Verdi</i>	<b>DUPLICATES</b>
0	«Henry»	<b>UNCHANGABLE</b>
1	«Ford»	<b>ORDERED</b>
2	«English»	

Tuple

<i>Verdi</i>	<b>NO-DUPLICATES</b>
«Henry»	<b>UNCHANGABLE</b>
«Ford»	<b>UNORDERED</b>
«English»	

Set

---

# Lists

`:::python`

<code>append()</code>	Adds an element at the end of the list
<code>clear()</code>	Removes all the elements from the list
<code>copy()</code>	Returns a copy of the list
<code>count()</code>	Returns the number of elements with the specified value
<code>extend()</code>	Add the elements of a list (or any iterable), to the end of the current list
<code>index()</code>	Returns the index of the first element with the specified value
<code>insert()</code>	Adds an element at the specified position
<code>pop()</code>	Removes the element at the specified position
<code>remove()</code>	Removes the item with the specified value
<code>reverse()</code>	Reverses the order of the list
<code>sort()</code>	Sorts the list

---

---

# Dictionaries

:::python	
clear()	Removes all the elements from the dictionary
copy()	Returns a copy of the dictionary
fromkeys()	Returns a dictionary with the specified keys and value
get()	Returns the value of the specified key
items()	Returns a list containing a tuple for each key value pair
keys()	Returns a list containing the dictionary's keys
pop()	Removes the element with the specified key
popitem()	Removes the last inserted key-value pair
setdefault()	Returns the value of the specified key. If the key does not exist: insert the key, with the specified value
update()	Updates the dictionary with the specified key-value pairs
values()	Returns a list of all the values in the dictionary

---

# Tuples

`:::python`

`count()` Returns the number of times a specified value occurs in a tuple

`index()` Searches the tuple for a specified value and returns the position of where it was found

---

# Sets

<code>add()</code>	Adds an element to the set
<code>clear()</code>	Removes all the elements from the set
<code>copy()</code>	Returns a copy of the set
<code>difference()</code>	Returns a set containing the difference between two or more sets
<code>difference_update()</code>	Removes the items in this set that are also included in another, specified set
<code>discard()</code>	Remove the specified item
<code>intersection()</code>	Returns a set, that is the intersection of two other sets
<code>intersection_update()</code>	Removes the items in this set that are not present in other, specified set(s)
<code>isdisjoint()</code>	Returns whether two sets have a intersection or not
<code>issubset()</code>	Returns whether another set contains this set or not
<code>issuperset()</code>	Returns whether this set contains another set or not
<code>pop()</code>	Removes an element from the set
<code>remove()</code>	Removes the specified element
<code>symmetric_difference()</code>	Returns a set with the symmetric differences of two sets
<code>symmetric_difference_update()</code>	inserts the symmetric differences from this set and another
<code>union()</code>	Return a set containing the union of sets
<code>update()</code>	Update the set with the union of this set and others

---

---

```
name = "Kristian"  
print(f"Hello, {name}!")  
print(f"{2*2}")  
print(f"Hello, {name.upper()}!")
```

```
overskudd = 500000.987654321  
print(f"Overskudd: {overskudd:.2f}")
```

# F-strings

```
university = {"name": "UiB", "location": "Bergen" }  
print(f"Enlisted at {university['name']}, campus {university['location']}")
```

---

# Dokumentasjon og hjelp

---



---

```
"""Gets and echo out a given string.
```

```
Parameters
```

```
-----
```

```
name : string
```

```
    A string that is part of a greeting
```

```
Returns
```

```
-----
```

```
string
```

```
    a string that contains a greeting
```

```
"""
```

```
# Kommentar
```

```
def echo_name(name: str) -> str:
```

```
    """Gets and echo out a given string.
```

```
Parameters
```

```
-----
```

```
name : string
```

```
    A string that is part of a greeting
```

```
Returns
```

```
-----
```

```
string
```

```
    a string that contains a greeting
```

```
"""
```

```
return(f"Hello {name}")
```

# Dokumentasjon

<https://docs.python.org/3/library/typing.html>

<https://peps.python.org/pep-0257/>

```
# Kommentar
```

```
def echo_name(name: str) -> str:
```

```
    return(f"Hello {name}")
```

---

# help()

```
>>> help(print)
>>> import random
>>> help(random)
>>> help(random.randint)
>>> help("if")
>>> help("symbols")
>>> help("keywords")
>>> help("modules")
```

Type "help", "copyright", "credits" or "license" for more information.

```
>>> help()
```

Welcome to Python 3.12's help utility! If this is your first time using Python, you should definitely check out the tutorial at <https://docs.python.org/3.12/tutorial/>.

Enter the name of any module, keyword, or topic to get help on writing Python programs and using Python modules. To get a list of available modules, keywords, symbols, or topics, enter "modules", "keywords", "symbols", or "topics".

Each module also comes with a one-line summary of what it does; to list the modules whose name or summary contain a given string such as "spam", enter "modules spam".

To quit this help utility and return to the interpreter, enter "q" or "quit".

---

---

Offisiell python dokumentasjon

<https://www.python.org/doc/>

Numpystyle docstrings

<https://numpydoc.readthedocs.io/en/latest/format.html#docstring-standard>

Realpython `help()`

<https://realpython.com/ref/builtin-functions/help/>

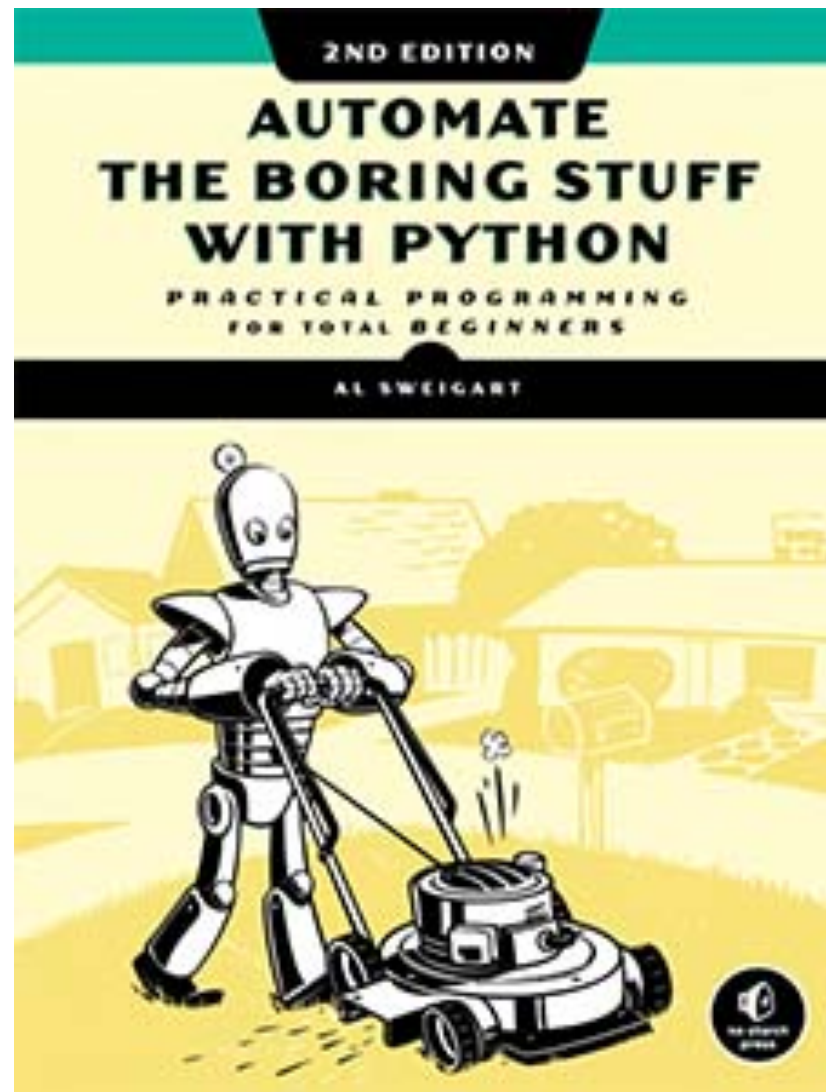
# Ressurser

Oppgavetid

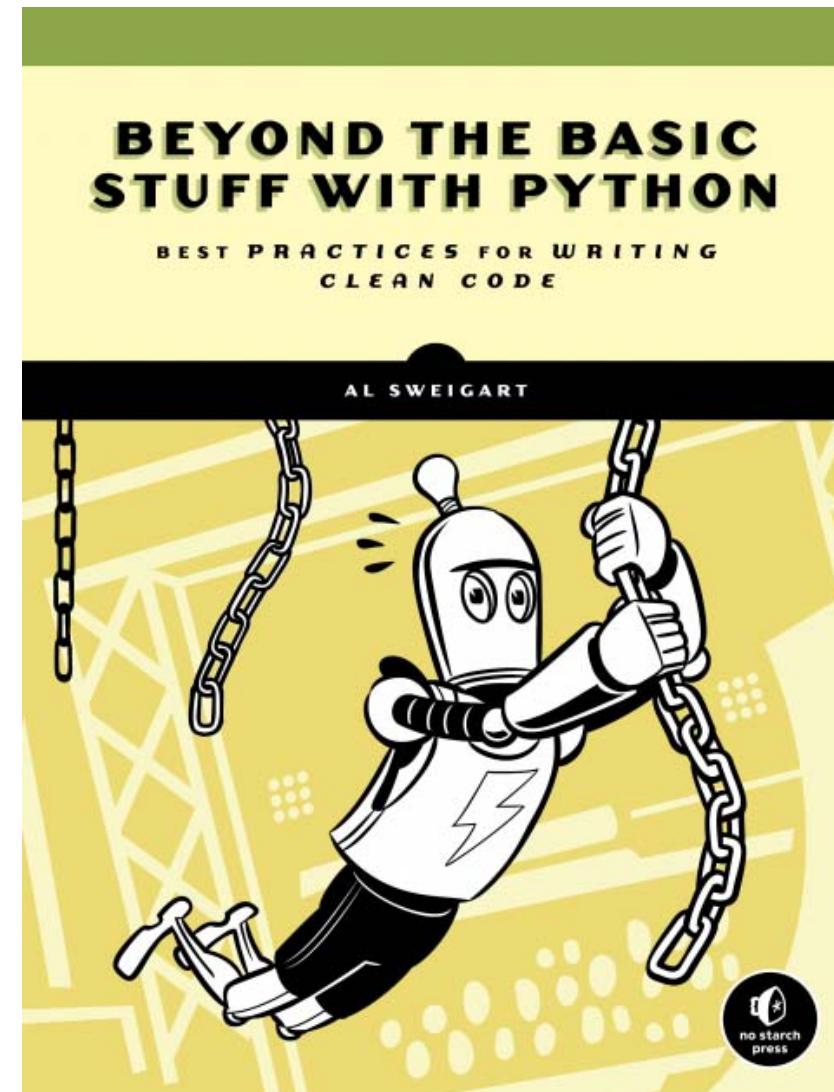
---

**Veien videre**

---



<https://automatetheboringstuff.com>



<https://inventwithpython.com/beyond/>



# Python

477,283 students



Overview



Practice



About Python

Practice Mode



352 contributors  
6239 mentors



## Explore the Python exercises on Exercism

Unlock more exercises as you progress. They're great practice and fun to do!

Search by title

All Exercises 140

Completed 0

In Progress 1

Available 139

Locked 0



### Hello World

In-progress

1 iteration

Exercism's classic introductory exercise. Just say "Hello, World!".



### Guido's Gorgeous Lasagna

Recommended

Learning Exercise

Learn the basics of Python by cooking Guido's Gorgeous Lasagna.



### Ghost Gobble Arcade Game

Available

Learning Exercise

Learn about booleans by setting up the rules for the Ghost Gobble arcade game.



### Currency Exchange

Available

Learning Exercise

Learn about numbers by solving Chandler's currency exchange conundrums.



### Meltdown Mitigation

Available

Learning Exercise

Learn about conditionals and avoid a meltdown by developing a simple control system for a Nuclear Reactor.



### Black Jack

Available

Learning Exercise

Learn about comparisons by implementing some Black Jack judging rules.



### Little Sister's Essay

Available

Learning Exercise

Learn about string methods while improving your little sister's school essay.



### Little Sister's Vocabulary

Available

Learning Exercise

Learn about strings by helping your little sister with her vocabulary homework.

<https://exercism.org/>

python

Norsk

Alle (171)

Programmering (155)

Vitenskap og teknologi (142)


Utdanning (81)


Studier og undervisning (53)

Spillutvikler (41)

Boter (38)

Samarbeid (37)







Python

We're a large community focused around the Python programming language.  
We believe that anyone can learn to code.

48 818 pålogget • 392 425 medlemmer





World of Coding | Programming, Code, Infosec, Hacking, Tech,  
Linux, Cybersec, Python, Java, math

WoC provides some of the best programming help on discord and provides a  
place for developers to find friends and relax!

8 299 pålogget • 63 003 medlemmer

<https://support.discord.com/>



- 
- [https://github.com/kbotnen/python\\_geovitenskap](https://github.com/kbotnen/python_geovitenskap)
  - <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/>
-