Avanceret Programmering (Uge 35)

Christian Gram Kalhauge (CKL)

Section 1

Introduktion

Agenda

- Introduktion
- Installering
- De første skridt i Python
- Evaluering

Lidt om mig

- Christian Gram Kalhauge
- 30 år
- 10 års programmerings erfaring
- Ph.D fra UCLA i Programmerings Sprog
- ckl@cphbuiness.dk

Hvem er I?

Corona

- 1 meters afstand imellem, 1 person per bord.
- 2 meters afstand fra mig.
- Sprit hænder af når i kommer.
- Ungå at dele materiale, Kuglepen, Computer.
- Bliv i lokalet, i fælleslokaler, eller uden for (ikke andre lokaler)
- Check Moodle.

Hvorfor Python?

- Interpreted Prototyping/Scripting Sprog
- Dynamically Typed
- Fantastisk std-lib og Extra Libraries.
- Ikke C#

De næste uger

- Vi skal blive meget nørded, men brugbart
- Tænk over et projekt som vil interesere dig:
    - Løse sudukoer?
    - En todo-app?
    - Et planlægnings værktøj?
    - En Skak computer?
    - Et tic-tac-toe spil?

Dansk Eller Engelsk?

- Dansk undervising
- Engelsk materiale og slides...

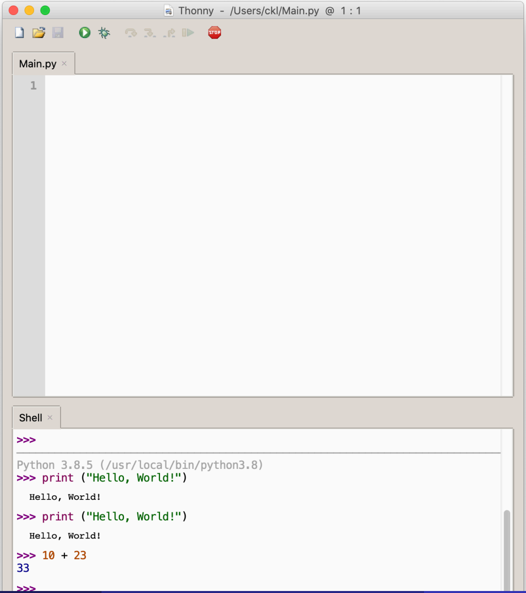
Section 2

Installering

Python3.8 og Thonny

- Python3.8: www.python.org
- Thonny: thonny.org

Thonny



```
>>>
Python 3.8.5 (/usr/local/bin/python3.8)
>>> print ("Hello, World!")
  Hello, World!
>>> print ("Hello, World!")
  Hello, World!
>>> 10 + 23
33
>>>
```

Section 3

De Første Skridt

Materiale

- A Byte of Python: https://python.swaroopch.com/
- Official Tutorial: https://docs.python.org/3/tutorial/index.html
- Std-library: https://docs.python.org/3/library/index.html

Syntax

```
# Comments
True, False # Booleans
1, 1.0  # Numbers
'c', 'Hello', "Hej" # Strings
""" Multiline
Strings """ # Many Lines Strings
```

Operators

```
>>> 1 + 2
3

>>> 'Hello,' + ' World!'
'Hello, World!'
```

Dynamically Typed

```
>>> 1 + ' World!'
Traceback (most recent call last):
  ...
TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

Dynamically Typed

```
>>> str(1) + ' World!'
'1 World!'
```

Dynamically Typed

```
>>> f'{1 + 2} World!'
'3 World!'
```

Read more about f-Strings

Variable

```
>>> x = 1
>>> x
1
```

IO and Functions

```
>>> print(True, 1, "hello")
True 1 hello

.>> s = input('What is your name? ')
What is your name? Christian
.>> s
'Christian'

.>> s = input('How old are you? ')
How old are you? 13
.>> int(s) # Read string as int
13
```

Assignment 1

1. Write a program that takes an first name and last name and print a greeting!
2. Calculate the birth year from your age

```
Hello, what is your name? Christian
Okay! And last name? Kalhauge
Yes, yes... And finally your age? 30
Very well, Christian Kalhauge,
  by my calculations you were born in 1990!
```

Lists

```
>>> y = [1, 2, 3]
>>> y[0]
1
>>> y[-1]
3
```

Lists (Append and Pop)

```
>>> y = [1, 2, 3]
>>> y.pop()
3
>>> y
[1, 2]
>>> y.append(10)
>>> y
[1, 2, 10]
```

Lists (Slices)

```
>>> y = [1, 2, 3]
>>> y[1:]
[2, 3]
>>> y[:-1]
[1, 2]
>>> y[1:-1]
[2]
```

Lists (Len)

```
>>> y = [1, 2]
>>> len(y)
2
```

Lists

But wait there is more:

- docs.python.org/3/tutorial/datastructures.html
- docs.python.org/3/library/stdtypes.html

Also works for strings

```
>>> y = "abcdefg"
>>> y[0:2]
'ab'
>>> y[2]
'c'
```

And tuples

```
>>> y = 1, 2, 3
>>> y
(1, 2, 3)
>>> y[0:2]
(1, 2)
>>> y[2]
3
```

Conditions

```
>>> if True:
...     print("It's true.") # four spaces
... elif not True:
...     print("It's false.")
... else:
...     print("It's neighter true or false.")
...
It's true.
```

While Loops

```
>>> i = 5
>>> while i > 0:
...     print(i)
...     i -= 1
...
5
4
3
2
1
```

For-each Loops

```
>>> lst = [ "a", "b", "c" ]
>>> for l in lst:
...     print(l)
...
a
b
c
```

For Loops

```
>>> for l in range(5):
...     print(l)
...
0
1
2
3
4
```

For Loops

```
>>> for l in range(4, -1, -1):
...     print(l)
...
4
3
2
1
0
```

For Loops

```
>>> for l in reversed(range(5)):
...     print(l)
...
4
3
2
1
0
```

Assignment 2

Create a reverse polish notation calculator:

```
+------- stack -------+
|                     | 7
| 7                   | 10
| 7, 10               | 20
| 7, 10, 20           | +
| 7, 30               | -
| -23                 |
```

1. Start with no stack feedback and only addition.
2. Extend with stack feedback, always 40 chars wide.
3. Handle too many elements to print on the stack.

Section 4

Evaluering