

Python (1) begynder

FOR IDA D. 12/03/2023

Omkring mig

- Anders Bensen Ottsen
 - ”Underviser” i dag
- Diplomingeniør i Softwareteknologi
 - Fra Syddansk Universitet
- (Næsten!) Civilingeniør i Computer Science & Engineering
 - Fra Danmarks Tekniske Universitet
- Arbejder pt. som Machine Learning Engineer ved Weel & Sandvig
 - Starter som Data Scientist ved PFA her 1. April
- Det her er 5. gang jeg underviser ”Python 1” for IDA

Omkring Jens

- Jens Kristian Vitus Bering
 - Fra IDA & hjælper med opgaver
- BSc. I Software Engineering
 - Fra Syddansk Universitet
- Læser cand.polyt i Software Engineering
 - Ved Syddansk Universitet
- Arbejder som studenterudvikler ved Bankdata
 - Hvor han programmerer finans software

Dagens program

- Variabler, betingelser og løkker
- Metoder, input og output
- Datastrukturer
- Programmer
- Hvad man kan gøre herfra og spørgerunde

Dagens program

- Variabler, betingelser og løkker
- Metoder, input og output
- Datastrukturer
- Programmer
- Hvad man kan gøre herfra og spørgerunde

Generelle programmeringsfærdigheder

Dagens program

- Variabler, betingelser og løkker
- Metoder, input og output
- Datastrukturer

Generelle programmeringsfærdigheder

- Programmer
- Hvad man kan gøre herfra og spørgerunde

Softwareudvikling

Dagens program

- Variabler, betingelser og løkker
- Metoder, input og output
- Datastrukturer

Generelle programmeringsfærdigheder

- Programmer

Softwareudvikling

- Hvad man kan gøre herfra og spørgerunde

Afrunding

Dagens program

- Variabler, betingelser og løkker
- Metoder, input og output
- Datastrukturer

Generelle programmeringsfærdigheder

- Programmer

Softwareudvikling

- Hvad man kan gøre herfra og spørgerunde

Afrunding

Lidt abstrakt

Dagens program

- Variabler, betingelser og løkker
- Metoder, input og output
- Datastrukturer
- Programmer
- Hvad man kan gøre herfra og spørgerunde

Vigtigst i dag

Generelle programmeringsfærdigheder

Softwareudvikling

Afrunding

Lidt abstrakt

Efter i dag kan I:

- Forstå generelle programmeringskoncepter
- Formulere og forstå simple Python programmer
- Køre et Python program I selv har skrevet
- Så hvordan når vi der til?
 - Ved små ”forelæsninger”
 - Med lidt liveprogrammering fra undertegnet
 - Opgaveløsning

Det online format

- Forelæsningerne bliver mig der taler i zoom
 - Bare unmute jer og tal løs hvis der er spørgsmål
- Ved opgaverne bliver I smidt ud i meeting rooms
 - Hvor i har ca. 30 minutter til at løse dem
 - Jens, Loc og jeg hopper rundt og hjælper i meeting rooms
 - ”Spørg efter hjælp” hvis i sidder fast! Så kommer vi (prøver i hvert fald)
 - I er selvfølgelig ikke tvunget til at tale med de andre, men det plejer at hjælpe!
- **TODO, LINK**

Programmering

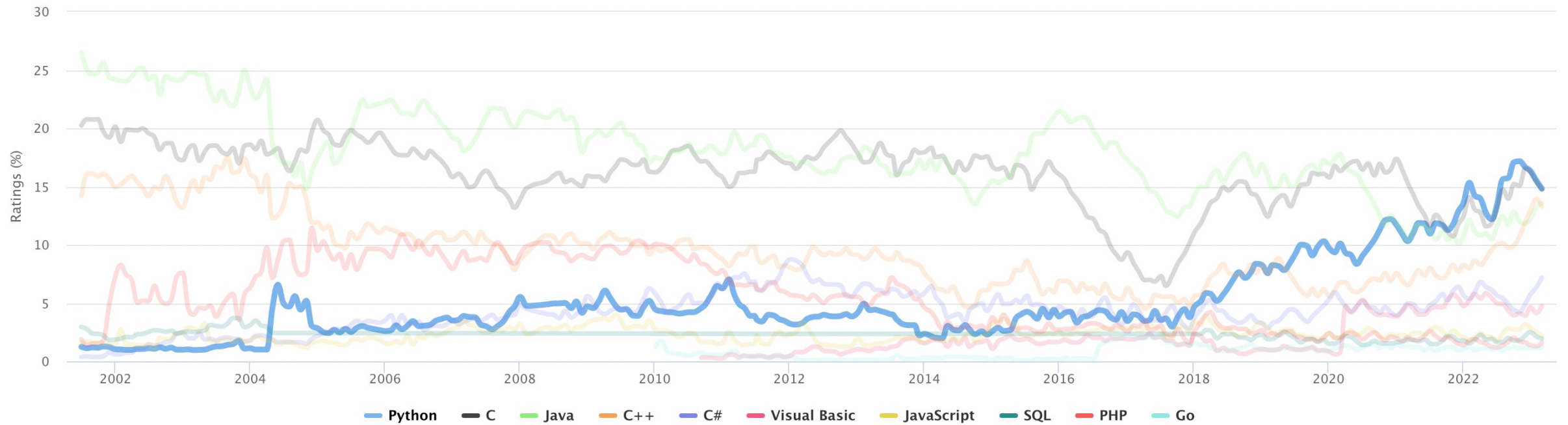
- Akten i at skrive ”koder” i et specifikt ”programmeringssprog” til en computer
 - Computeren oversætter det så til sit eget ”sprog”
 - Som så til sidst bliver eksekveret i ”hardware”
- Vi kommunikere altså med computeren (og dermed hardware) igennem et sprog
 - Og får den til at udføre opgaver for os

Python

- Et multi-paradigme programmeringssprog
 - Så hvad betyder det?
 - Paradigme → en speciel måde at anskue tingene på
- Opfundet i 1991
- Et af verdens mest populære programmeringssprog

TIOBE Programming Community Index

Source: www.tiobe.com



Hello World

- Det første program alle lærer at lave!

Hello World

- Det første program alle lærer at lave!

- I python



```
1  
2  print("hello world")  
3
```


Hello World

- Det første program alle lærer at lave!

Kode linjer

- I python



```
1  
2 print("hello world")  
3
```

Hello World

- Det første program alle lærer at lave!

- I python



Kode linjer En metode

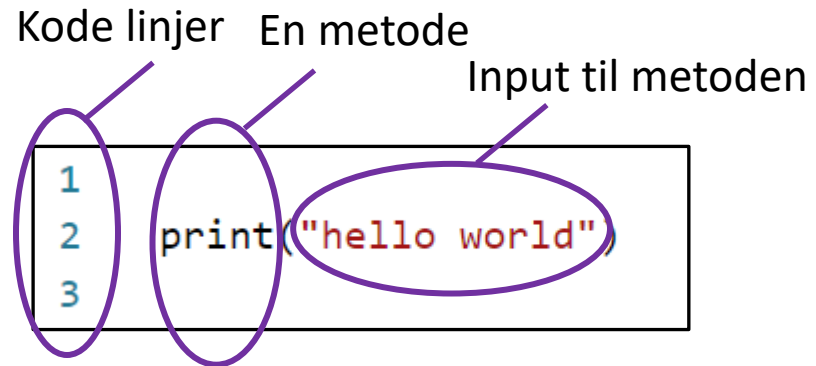


```
1  
2  
3 print("hello world")
```

Hello World

- Det første program alle lærer at lave!

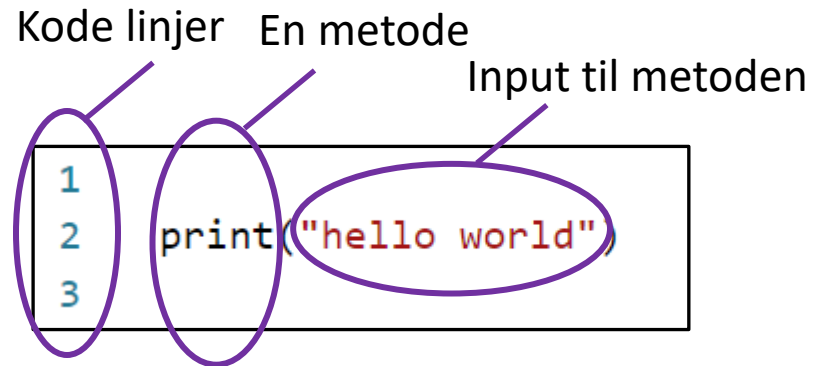
- I python



Hello World

- Det første program alle lærer at lave!

- I python



- Total forvirring, lad os se det live!

Jeres (måske) første (Python) program

- Alle åbner Visual Studio Code (VS code, eller bare code)

VARIABLER, BETINGELSER & LØKKER

Variabler

- Lidt som i kender det fra matematik
- Vi kan i computerens verden gemme information/data i variabler
- Som f.eks.: $x = 2$
 - Eller: `president_of_usa = "Joe Biden"`
 - Har tidligere været: `president_of_usa = "Donald Trump"` osv.
 - Så variablers indhold kan godt ændres

Variabler i computere

- Mange forskellige typer af variabler. De mest kendte er:
- Heltal (integers)
 - 1, 256, 712371273
- Kommatal (floats)
 - 1.0, 3.1415, 359.12391293
- Streng (strings)
 - "Anders", "Python", "123123213144", "1.39"
- Characters (chars)
 - 'a', 'b', 'c'
- Booleans
 - True, False

Variabler i computere

- Mange forskellige typer af variabler. De mest kendte er:

- Heltal (integers)

- 1, 256, 712371273

- Kommatal (floats)

- 1.0, 3.1415, 359.12391293

- Streng (strings)

- "Anders", "Python", "123123213144", "1.39"

- ~~• Characters (chars)~~

~~a', 'b', 'c'~~



Characters er en reel ting i computere (og LANGT de fleste programmeringssprog), men ikke i Python, så den taler vi ikke mere om nu!

- Booleans

- True, False

Variabler i Python

```
1  
2     age = 25  
3     height = 1.72  
4     name = "Anders"  
5     likes_python = True  
6
```

Tal og matematik

- Man kan bruge matematiske operationer på tal i programmering

<i>Name</i>	<i>Meaning</i>	<i>Example</i>	<i>Result</i>
+	Addition	$34 + 1$	35
-	Subtraction	$34.0 - 0.1$	33.9
*	Multiplication	$300 * 30$	9000
/	Division	$1.0 / 2.0$	0.5

Tal og assignments i Python

```
1  current_age = 25
2  age_in_25_years = current_age*2 #50
3  age_12_5_years_ago = current_age/2 #12.5
4  anders_height = 1.72
5  jens_height = anders_height + 0.10 #1.82
6  first_name = "Anders"
7  last_name = "Ottsen"
8  full_name = first_name + " " + last_name #"Anders Ottsen"
```

- Så vi kan altså bruge variabler til at beskrive andre variabler
- Mht. parenteser

Increment/decrement

```
1  age = 25
2  age -= 1 #25-1 = 24
3  age += 1 #24+1 = 25
4  age = age-1 #25-1 = 24
5  age = age+1 #24+1 = 25
```

Booleans

- Skal som sagt enten være true eller false
- Meget vigtig til at beskrive betingelser
- Relationelle operatorer:

<i>Operator</i>	<i>Mathematics Symbol</i>	<i>Name</i>	<i>Example (radius is 5)</i>	<i>Result</i>
<	<	less than	<code>radius < 0</code>	<code>false</code>
<=	≤	less than or equal to	<code>radius <= 0</code>	<code>false</code>
>	>	greater than	<code>radius > 0</code>	<code>true</code>
>=	≥	greater than or equal to	<code>radius >= 0</code>	<code>true</code>
==	=	equal to	<code>radius == 0</code>	<code>false</code>
!=	≠	not equal to	<code>radius != 0</code>	<code>true</code>

Booleans i Python

```
1  jens_age = 26
2  anders_age = 25
3  whos_oldest_1 = jens_age > anders_age # True
4  whos_oldest_2 = jens_age < anders_age # False
5  whos_oldest_3 = jens_age >= anders_age # True
6  whos_oldest_4 = jens_age <= anders_age # False
7  is_age_equal = jens_age == anders_age # False
8  is_age_nequal = jens_age != anders_age # True
```

Konvertering af variabler

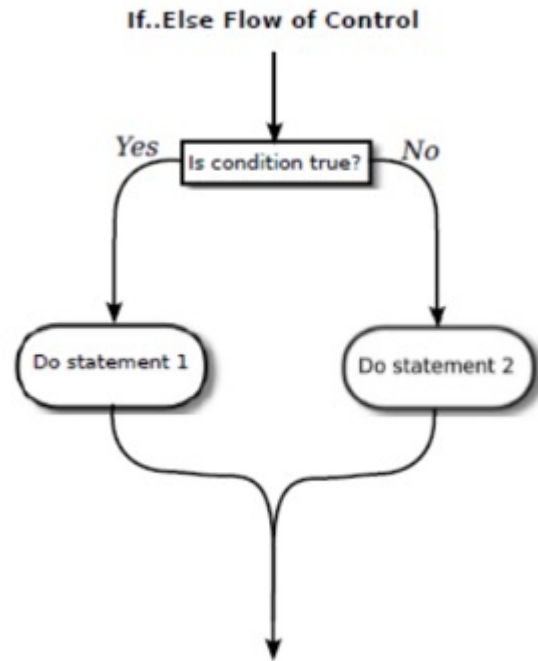
```
1 age = 25
2 age = str(age) # '25'
3 age = float(age) # 25.0
4 age = int(age) # 25
```

- Vi kan altså ændre på variabel typen til en anden!

Betingelser

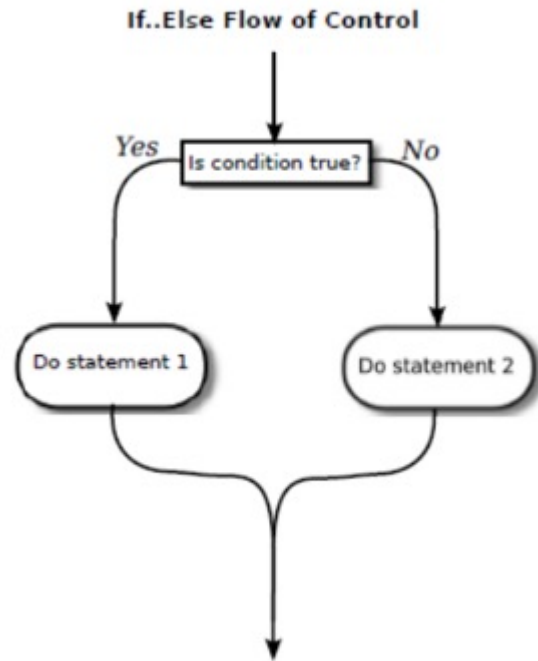
- Bruges til at beskrive ”flowet” i et program
- Bruger booleans til at bestemme hvad programmet skal gøre

Betingelser



```
if ( boolean_expression ):  
    statement  
else:  
    statement
```

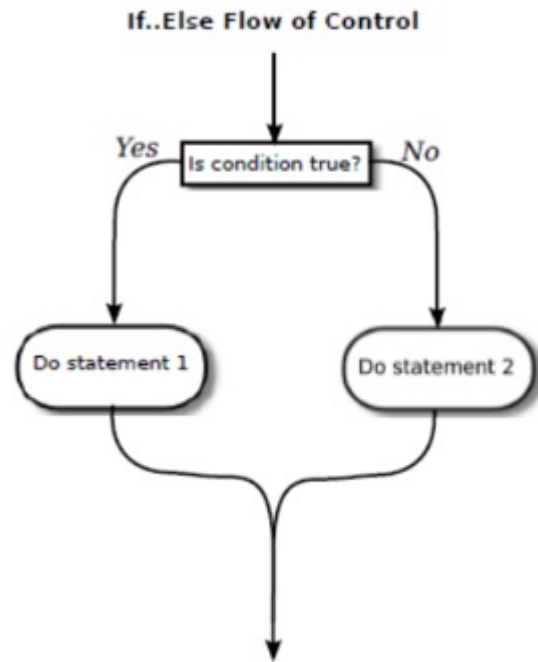
Betingelser



```
if ( boolean_expression ):  
    statement  
else:  
    statement
```

- Mht. indryk

Betingelser



```
if ( boolean_expression ):  
    statement  
else:  
    statement
```

- Mht. indryk
- Lad os tage et live eksempel med Jens' og min alder igen

Løkker

- Bruges til at gentage noget kode flere gange
- Er meget grundlæggende inden for programmering
- Der er flere typer. I dag beskæftiger vi os med to forskellige:
 - While løkken
 - For løkken

Løkker

100 times {
 print("Hello World!")
 print("Hello World!")
 print("Hello World!")
 ...
 ...
 print("Hello World!")

- Hvis vi nu gerne vil lave ovenstående kan vi gøre det med en løkke
- Motivation → Hvorfor skrive det manuelt 100 gange, hvis vi kan skrive noget kode der gør det for os?

While løkken

- While løkken udfører noget så længe at dens betingelse er sand
- Dens basis struktur er sådan her:

```
1  i = 0
2  while (i < 10):
3      print("Hello World!")
4      i += 1
```

While løkken

- While løkken udfører noget så længe at dens betingelse er sand
- Dens basis struktur er sådan her:

```
1  i = 0
2  while (i < 10):
3      print("Hello World!")
4      i += 1
```

- Lad os tage den live!

For-løkken

- En lidt anden slags løkke end while-løkken
 - Men de kan cirka de samme ting!
- Dens basis struktur er sådan her:

```
1  for i in range(10):  
2  |      print("Hello World!")
```

For-løkken

- En lidt anden slags løkke end while-løkken
 - Men de kan cirka de samme ting!
- Dens basis struktur er sådan her:

```
1  for i in range(10):  
2  |      print("Hello World!")
```

- Lad os tage den live!

Opgaver pt. 1 (30 minutter)

- 1. Lav et program som kan printe: Hej + "dit navn" 10 gange. Når den har printet det 10 gange, så skal den printe: Jeg elsker + "din yndlingsøl" 10 gange.**
 - Hint: For løkke med betingelser
- 2. Lav et program som kan regne 20 grader celsius om til fahrenheit og herefter printe resultatet. Når resultatet er printet skal programmet regne det tilbage til celsius, og printe det resultat.**
 - Formlen er: $\text{fahrenheit_degrees} = (1.8 * \text{celsius_degrees}) + 32$
 - Formlen er: $\text{celsius_degrees} = (\text{fahrenheit_degrees} - 32) / 1.8$
- 3. (Ekstra) Vi har 2 punkter på et koordinatsystem: A(2,7) og B(7,9). Lav et program som kan regne afstanden fra A til B (kaldes også $d(A,B)$) og print den.**
 - Formlen er: $\text{distance} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$
 - Hint: Brug `math.sqrt()` for kvadratrodd og `**` for potens i python
 - Husk at skrive "import math" øverst i filen
 - Ekstra: Vi har også et punkt C(7,2). Udregn afstanden $d(B,C)$ og få programmet til at printe fortælle hvilken afstand er størst.

METODER, INPUT & OUTPUT

Metoder

- Indtil videre har vi set hvordan vi laver mindre programmer som bare kører direkte i en fil
 - Det her er ikke en synderlig holdbar løsning
 - Den her fil kan blive virkelig lang ved større programmer
- Derfor metoder
 - Gem noget funktionalitet ud i en blok
- I har allerede stødt på en metode, nemlig:
 - `print(`)
 - Metoden *print* tager en streng som input, og står så for at printe det vi vil have til konsollen

Metoder


- Lad os tage et eksempel:

```
1  def print_name():  
2      print("anders")  
3  
4  print_name()
```

Metoder

- Lad os tage et eksempel:

Vi "definerer" en
metode



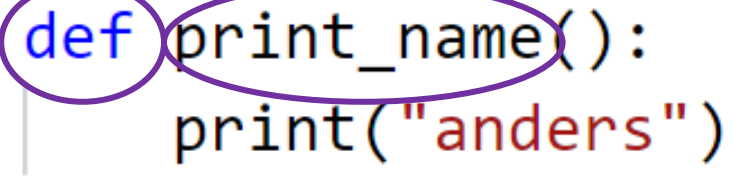
```
1 def print_name():  
2     print("anders")  
3  
4 print_name()
```

Metoder

- Lad os tage et eksempel:

Vi "definerer" en
metode

Metodens navn



```
1 def print_name():  
2     print("anders")  
3  
4 print_name()
```

The code is enclosed in a black rectangular box. The word 'def' on line 1 is circled in purple, with a line pointing to the text 'Vi "definerer" en metode'. The text 'print_name()' on line 1 is also circled in purple, with a line pointing to the text 'Metodens navn'.

Metoder

- Lad os tage et eksempel:

Vi "definerer" en
metode

Metodens navn

```
1  def print_name():  
2      print("anders")  
3  
4  print_name()
```

Metodens krop

Metoder

- Lad os tage et eksempel:

Vi "definerer" en
metode

Metodens navn

```
1  def print_name():
2      print("anders")
3
4  print_name()
```

Metodens krop

Metode kald

Metoder

```
1  def print_name():  
2      print("anders")  
3  
4  print_name()
```

- Hvad nu hvis vi vil have at vores program kan printe andre navne?
 - Vi kan bruge et argument!

Metoder

```
1  def print_name(name):  
2      print(name)  
3  
4  print_name("Anders")
```

Metoder

```
1  def print_name(name):  
2      |    print(name)  
3  
4  print_name("Anders")  
5  print_name("Jens")
```

- Nu har vi lige pludselig noget kode som er genbrugeligt!
 - Lad os se den live

Metoder og returnering

- Metoder behøver ikke bare at printe deres argument
 - De kan også ”returnere ting”
- De kan returnere alle de data typer vi har talt om
 - Integer, float, string, boolean mm.
 - Men også datastrukturer
 - Mere om det senere

Metoder og returnering

- Vores mål: Lav en metode som returnerer det største af to tal

Metoder og returnering

- Vores mål: Lav en metode som returnerer det største af to tal

```
1  def get_biggest_number(x1, x2):  
2      if (x1 > x2):  
3          return x1  
4      else:  
5          return x2  
6  
7  x = 10  
8  y = 20  
9  biggest_number = get_biggest_number(x,y)
```


Python metoder

- Python kommer med mange indbyggede metoder
 - Som kan bruges til mange forskellige ting!
- F.eks. Indeholder "math" "biblioteket" mange smarte metoder
- Og den indbyggede metode "input" gør at man kan skrive med sit program i konsollen

Math ”biblioteket”

- I python kan vi importere et ”bibliotek”
 - Det er sådan set bare en fil fyldt med metoder vi så kan bruge

```
1  import math
2
3  my_int = 100
4
5  sqrt_my_int = math.sqrt(my_int) #10
6
7  log2_my_int = math.log2(my_int) #6.64385.....
```

Math ”biblioteket”

- I python kan vi importere et ”bibliotek”
 - Det er sådan set bare en fil fyldt med metoder vi så kan bruge

```
1 import math
2
3 my_int = 100
4
5 sqrt_my_int = math.sqrt(my_int) #10
6
7 log2_my_int = math.log2(my_int) #6.64385.....
```

random ”biblioteket”

```
1  import random
2
3  random.randint(1, 20) # random tal mellem 1 til 20
```

Input metoden

- Indtil videre har vi skrevet programmer der ikke ændrer sig efter de er kørt
 - Hvad nu hvis vi gerne vil give input løbende?
- Pythons "input" metode kan bruges til at modtage strings, integers osv.
 - Man skriver i konsollen, så tager metoden og bruger det input i programmet
- Lad os se et eksempel

Input metoden

```
1  print("Please enter your name: ")
2  name = input()
3  print("Please enter your age: ")
4  age = input()
5  print("Hello " + name + ", you are: " + age + " years old")
```

Input metoden

```
1  print("Please enter your name: ")
2  name = input()
3  print("Please enter your age: ")
4  age = input()
5  print("Hello " + name + ", you are: " + age + " years old")
```

- Lad os se den live

Input metoden

- Hvad nu hvis vi ikke vil have at programmet stopper efter 1 gang?
 - Vi bruger da en løkke!

Input metoden

- Hvad nu hvis vi ikke vil have at programmet stopper efter 1 gang?
 - Vi bruger da en løkke!

```
1  while (True):  
2      print("Please enter your name: ")  
3      name = input()  
4      print("Please enter your age: ")  
5      age = input()  
6      print("Hello "+ name + ", you are: " + age + " years old")
```

Input metoden

- Hvad nu hvis vi ikke vil have at programmet stopper efter 1 gang?
 - Vi bruger da en løkke!

```
1  while (True):  
2      print("Please enter your name: ")  
3      name = input()  
4      print("Please enter your age: ")  
5      age = input()  
6      print("Hello "+ name + ", you are: " + age + " years old")
```

- Igen, live!

Opgaver pt. 2 (30 minutter)

1. Lad os bygge videre på opgave 1.2 fra tidligere – Lav selve udregningen fra celsius til fahrenheit om til en metode ”def celsius_to_fahrenheit(celsius_degrees)”. Brug herefter så input metoden så det er brugeren der fortæller hvilke grader der skal regnes. Print fahrenheit graderne til sidst.
 - Husk at konvertere inputtet fra brugeren til int!
2. Lav et program som kan udregne din promille. Brug input metoden så du kan skrive hvor mange genstande du har drukket.
 - Formlen er: $promille = \frac{amount_of_units \cdot 12}{body_weight \cdot 0.7}$, lav gerne denne til en metode
 - Brug et loop til at få dit program til at starte forfra når den har udregnet promillen
3. (Ekstra) Lav en metode ”def is_equal(number)” som kan tjekke om et givet tal er lige (Return true) eller ulige (Return false)
 - Hint:
 - $4\%2 = 0$
 - $5\%2 = 1$

DATASTRUKTURER

Datastrukturer

- Så nu har vi lært lidt basale principper
 - Og lært hvordan man kan gemme data i en variabel
- Men hvad nu hvis vi gerne vil gemme en ”samling” af data
 - F.eks. Hvis vi skulle lave et program der holdte alle CPR numre for danskere
 - Vi gider ikke gemme en variabel for hver eneste CPR nummer
 - Så skulle vi manuelt skrive ~6 millioner variabler.....
 - Vi bruger i stedet en datastruktur!

Datastrukturer

- Definition af en datastruktur:
 - Noget data der er organiseret i elementer, hvor man kan tilføje og fjerne data fra den struktur
- Definitionen er kludret, men ideen er simpel!
- Når man skal ud og handle laver man en indkøbsliste
 - Den her liste består af varer vi skal købe
 - Det er det samme med en datastruktur! Vi kan lave en liste af variabler

Lister

- En liste er en datastruktur hvor man kan gemme flere variabler i en variabel
 - Man kan tilføje fra en liste og fjerne fra en liste
- Man kan bygge en liste af de variabel typer vi allerede kender
 - Så en liste af integers er bare en variabel der holder flere heltal

Lister

- Vi starter med en tom liste

Lister

- Og tilføjer så en integer 10

10

Lister

- Og tilføjer så en integer 8

10	8
----	---

Lister

- Og tilføjer så en integer 17

10	8	17
----	---	----

Lister

- Og tilføjer så en integer 17

10	8	17
----	---	----

- Så nye elementer ryger bare bagerst i listen efter tilføjelse

Lister

- Vi fjerner så 8

10	17
----	----

Lister

- Vi fjerner så 8

10	17
----	----

- Så når man fjerner en variabel bliver de resterende elementer skubbet frem

List i Python

```
1 integer_list = []
2 integer_list.append(10) # 10
3 integer_list.append(8) # 10,8
4 integer_list.append(17) # 10,8,17
5
6 integer_list.remove(8) # 10,17
7
8 len(integer_list) # 2
```

List i Python

```
1 integer_list = []
2 integer_list.append(10) # 10
3 integer_list.append(8) # 10,8
4 integer_list.append(17) # 10,8,17
5
6 integer_list.remove(8) # 10,17
7
8 len(integer_list) # 2
```

- Live!

Løkker og lister

```
1 integer_list = [1,3,7,2,5]
2
3 for i in integer_list:
4     | print(i)
```

Dictionaries

- En anden slags datastruktur!
- Så nu har vi lært om lister
 - Det er nok den mest basale datastruktur
 - Men det er kun en af mange forskellige datastrukturer
- Et dictionary (også kaldet et map) fungerer præcis som en ordbog
 - Når man slår et ord op så søger man på ordet og så kommer der en beskrivelse af ordet
 - I et dict (dictionary) har vi en key, der mapper til en value
 - Så ligesom en ordbog.....

Dictionaries i python

```
1  my_dict = {}
2
3  my_dict['Beer'] = 'An alcoholic beverage brewed on hops'
4  my_dict['Soda'] = 'Water with sugar'
5  my_dict['Jens'] = 'Handsome dude from IDA'
6
7  my_dict.get('Beer') # An alcoholic beverage brewed on hops
8  my_dict.values() # ['An alcoholic...', 'Water with...', 'Handsome..']
9  my_dict.keys() # ['Beer', 'Soda', 'Jens']
```

Dictionaries i python

```
1  my_dict = {}
2
3  my_dict['Beer'] = 'An alcoholic beverage brewed on hops'
4  my_dict['Soda'] = 'Water with sugar'
5  my_dict['Jens'] = 'Handsome dude from IDA'
6
7  my_dict.get('Beer') # An alcoholic beverage brewed on hops
8  my_dict.values() # ['An alcoholic...', 'Water with...', 'Handsome..']
9  my_dict.keys() # ['Beer', 'Soda', 'Jens']
```

- Live!

Opgaver pt. 3 (30 minutter)

1. Lav et program som fylder en liste med 20 tilfældige tal mellem 1 til 100 og print den til sidst
 - Hints:
 - Brug metoden: `randint(min,max)` til at generere tilfældige tal i et interval
 - Husk at skrive `import random` i toppen af jeres program
 - Brug så en løkke der kører 20 gange hvor `randint` bliver kaldt
2. Lav et program som udregner gennemsnittet af en liste med tal
 - Hints:
 - Løb igennem listen og plus hver variabel sammen
 - Eller brug metoden `sum()` som returnerer summen af en liste
3. (Ekstra) Lav et program der fylder et map(dict) med 20 random tal mellem 1 og 100 (mappets values) hvor key er tallet fra 0...19
 - Udregn herefter gennemsnittet af alle valuesne (de random tal) og print det til sidst
4. (Ekstra, svær) Vi har to 3d vektorer $a=[a_1,a_2,a_3]$, $b=[b_1,b_2,b_3]$ lav et program som fylder deres værdier med tilfældige tal og udregn herefter prikproduktet mellem dem: $a_1*b_1+a_2*b_2+a_3*b_3$

PROGRAMMER

Programmer

- Nu har vi faktisk lært de mest basale programmeringskoncepter!
 - Variabler
 - Betingelser
 - Løkker
 - Datastrukturer
- Alle de her ting er fælles for nærmest alle programmeringssprog, og er meget generelle koncepter
 - Og nogle af de her principper er sådan set bare matematik, som daterer tilbage før vi overhovedet havde en computer

Programmer

- Når man laver et program bruger man alle de principper vi har lært
 - Men man programmerer efter et formål/ide
- Alle de ting i bruger i jeres computere bruger de her basale programmeringskoncepter, om det er word, jeres operativsystem, facebook osv.

Et konkret program

- Case: En basal kalender
 - Lad os se den live

Afrunding & anbefalinger

- Programmering er et håndværk
 - Man lærer det ved at gøre det
- Tag den videre her fra:
 - <https://www.learnpython.org/>
 - <https://www.w3schools.com/python/>
 - <https://www.codecademy.com/catalog/language/python>
- Alle løsninger til opgaverne og kalenderen:
 - https://github.com/AndersBensen/python_101/raw/main/python1/exercises.zip

Spørgsmål?

- anders_bensen@hotmail.com