Python Lecture 1

March 21, 2016

1 Tal

```
In [93]: 2+2
```

```
Out[93]: 4
```

```
In [94]: # This is a comment
```

```
In [95]: 2+2  # and a comment on the same line as code
```

```
Out[95]: 4
```

1.1 Uttryck

```
In [96]: (50-5*6)/4
```

```
Out[96]: 5.0
```

Heltalsdivision avrundar nedåt

```
In [97]: 7/3
```

```
Out[97]: 2.3333333333333335
```

```
In [98]: 7/-3
```

```
Out[98]: -2.3333333333333335
```

2 Variabler

```
In [99]: width=20
         print(width)
```

```
20
```

```
In [100]: height=5*9
          print(height)
```

```
45
```

```
In [101]: width*height
```

```
Out[101]: 900
```

```
In [102]: x = y = z = 0 # Tilldela x, y och z 0
```

```
In [103]: x
```

```
Out[103]: 0
```

```
In [104]: y
```

```
Out[104]: 0
```

```
In [105]: z
```

```
Out[105]: 0
```

3 Flyttal

```
In [106]: 3*3.75/1.5
```

```
Out[106]: 7.5
```

```
In [107]: 7.0/2
```

```
Out[107]: 3.5
```

4 Teckensträngar

```
In [108]: 'spam eggs'
```

```
Out[108]: 'spam eggs'
```

```
In [109]: 'doesn\'t'
```

```
Out[109]: "doesn't"
```

```
In [110]: "doesn't"
```

```
Out[110]: "doesn't"
```

```
In [111]: '"Yes," he said.'
```

```
Out[111]: '"Yes," he said.'
```

```
In [112]: "\"Yes,\" he said."
```

```
Out[112]: '"Yes," he said.'
```

```
In [113]: '"Isn\'t," she said.'
```

```
Out[113]: '"Isn\'t," she said.'
```

5 Teckensträngar

```
In [114]: hello = "This is a rather long string containing\n several lines of text just as you would do
              Note that whitespace at the beginning of the line is significant."
```

```
In [115]: print(hello)
```

```
This is a rather long string containing
 several lines of text just as you would do in C.
    Note that whitespace at the beginning of the line is significant.
```

6 Teckensträngar

```
In [116]: hello = r"This is a rather long string containing\n\
          several lines of text much as you would do in C."

In [117]: print(hello)

This is a rather long string containing\nseveral lines of text much as you would do in C.
```

7 Teckensträngar

```
In [118]: print("""
          Usage: thingy [OPTIONS]
               -h                        Display this usage message
               -H hostname               Hostname to connect to
          """)

Usage: thingy [OPTIONS]
     -h                        Display this usage message
     -H hostname               Hostname to connect to
```

8 Teckensträngar

```
In [119]: word = 'Help' + 'A'

In [120]: word

Out[120]: 'HelpA'

In [121]: '<' + word*5 + '>'

Out[121]: '<HelpAHelpAHelpAHelpAHelpA>'

In [122]: word[4]

Out[122]: 'A'

In [123]: word[0:2]

Out[123]: 'He'

In [124]: word[2:4]

Out[124]: 'lp'
```

9 Teckensträngar

```
In [125]: word[:2] # Första 2 tecknen

Out[125]: 'He'

In [126]: word[2:] # Alla utom de första 2 tecknen

Out[126]: 'lpA'
```

10 Teckensträngar

```
In [127]: word[0] = 'x'
```

```
        ---------------------------------------------------------------------------

        TypeError                                 Traceback (most recent call last)

        <ipython-input-127-d89b9f9f38d7> in <module>()
    ----> 1 word[0] = 'x'


        TypeError: 'str' object does not support item assignment
```

```
In [207]: word[:1] = 'Splat'
```

```
        ---------------------------------------------------------------------------

        TypeError                                 Traceback (most recent call last)

        <ipython-input-207-f21641443054> in <module>()
    ----> 1 word[:1] = 'Splat'


        TypeError: 'str' object does not support item assignment
```

11 Teckensträngar

```
In [208]: word[-1] # Sista tecknet

Out[208]: 'A'

In [209]: word[-2] # Näst sista tecknet

Out[209]: 'p'

In [210]: word[-2:] # De sista två tecknen

Out[210]: 'pA'

In [211]: word[:-2] # Alla utom de två sista tecknen

Out[211]: 'Hel'

In [212]: s = 'supercalifragilisticexpialidocious'

In [213]: len(s)

Out[213]: 34
```

12 Listor

- Sammansatt datatyp
- Används för att gruppera ihop värden
- Kan innehålla olika datatyper
- Börjar med index 0
- Individuella värden kan ändras

13 Lista

```
In [214]: a = ['spam', 'eggs', 100, 1234]

In [215]: a

Out[215]: ['spam', 'eggs', 100, 1234]

In [216]: a[0]

Out[216]: 'spam'

In [217]: a[3]

Out[217]: 1234

In [218]: a[-2]

Out[218]: 100

In [219]: a[1:-1]

Out[219]: ['eggs', 100]

In [220]: a[:2] + ['bacon', 2*2]

Out[220]: ['spam', 'eggs', 'bacon', 4]

In [221]: 3*a[:3] + ['Boe!']

Out[221]: ['spam', 'eggs', 100, 'spam', 'eggs', 100, 'spam', 'eggs', 100, 'Boe!']
```

14 Lista

Ersätta värden

```
In [222]: a[0:2] = [1,12]

In [223]: a

Out[223]: [1, 12, 100, 1234]
```

Ta bort värden

```
In [224]: a[0:2] = []

In [225]: a

Out[225]: [100, 1234]
```

Infoga värden

```
In [226]: a[1:1] = ['bletch', 'xyzzy']

In [227]: a

Out[227]: [100, 'bletch', 'xyzzy', 1234]

In [228]: a[:0] = a

In [229]: a

Out[229]: [100, 'bletch', 'xyzzy', 1234, 100, 'bletch', 'xyzzy', 1234]
```

15 Lista

```
In [230]: len(a)

Out[230]: 8
```

Listor i listor

```
In [231]: q = [2,3]

In [232]: p = [1, q, 4]

In [233]: len(p)

Out[233]: 3

In [234]: p[1]

Out[234]: [2, 3]

In [235]: p[1][0]

Out[235]: 2

In [236]: p[1].append('extra')

In [237]: p

Out[237]: [1, [2, 3, 'extra'], 4]

In [238]: q

Out[238]: [2, 3, 'extra']
```

16 Programmering

Exempel: Fibonnaci serie. Summan av två element definiera nästa tal:

```
In [239]: a, b = 0, 1
          while b < 100:
              print(b)
              a, b = b, a+b
```

```
1
1
2
3
5
8
13
21
34
55
89
```

17 Programmering utskrift

```
In [240]: i = 256*256
```

```
In [241]: print('The value of i is', i)
```

```
The value of i is 65536
```

```
In [242]: a, b = 0, 1
          while b < 1000:
              print(b, end=" ")
              a, b = b, a+b
```

```
1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987
```

18 Flödeskontroll

```
In [243]: x = 56
          if x < 0:
              x = 0
              print('Negative changed to zero')
          elif x == 0:
              print('Zero')
          elif x == 1:
              print('Single')
          else:
              print('More')
```

```
More
```

19 for-loopar

Skriva ut längden på strängar

```
In [244]: a = ['cat', 'window', 'defenestrate']
          for x in a:
              print(x, len(x))
```

```
cat 3
window 6
defenestrate 12
```

20 range() funktionen

```
In [245]: for x in range(10):
              print(x)
```

```
0
1
2
3
4
5
6
7
8
9
```

```
In [246]: for x in range(5,10):
              print(x)
```

```
5
6
7
8
9
```

```
In [247]: for x in range(0,10,3):
              print(x)
```

```
0
3
6
9
```

```
In [248]: for x in range(-10,-100,-30):
              print(x)
```

```
-10
-40
-70
```

21 range() funktionen

```
In [249]: a = ['Mary', 'had', 'a', 'little', 'lamb']
          for i in range(len(a)):
              print(i, a[i])
```

```
0 Mary
1 had
2 a
3 little
4 lamb
```

22 break, continue och else i loopar

- break

- Avbryter en loop
- continue
- Fortsätter till nästa iteration
- else
- Anropas när loopen är slut, men inte om break använts

23 break, continue och else i loopar

```
In [250]: for n in range(2, 10):
              for x in range(2, n):
                  if n % x == 0:
                      print(n, 'equals', x, '*', n/x)
                      break
              else:
                  # loop fell through without finding a factor
                  print(n, 'is a prime number')
```

```
2 is a prime number
3 is a prime number
4 equals 2 * 2.0
5 is a prime number
6 equals 2 * 3.0
7 is a prime number
8 equals 2 * 4.0
9 equals 3 * 3.0
```

pass-satser
Ibland kan det vara bra att skapa kod-skelett utan implementering. Detta kan göras med pass-satsen:

```
while True:
    pass
```

24 Funktioner

Funktioner i Python definieras med def-satsen:

```
In [251]: def fib(n):
              a, b = 0, 1
              while b < n:
                  print(b, end=" ")
                  a, b = b, a+b
```

```
In [252]: fib(2000)
```

```
1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597
```

25 Funktioner retur-värde

```
In [253]: def fib2(n):
              result = []
              a, b = 0, 1
              while b < n:
                  result.append(b)
                  a, b = b, a+b
              return result
```

```
In [254]: f100 = fib2(100)

In [255]: f100

Out[255]: [1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]
```

26 Standardargument

```
In [256]: def ask_ok(prompt, retries=4, complaint='Yes or no, please!'):
              while True:
                  ok = input(prompt)
                  if ok in ('y', 'ye', 'yes'): return 1
                  if ok in ('n', 'no', 'nop', 'nope'): return 0
                  retries = retries - 1
                  if retries == 0:
                      print('Giving up')
                      return
                  print(complaint)

In [261]: ask_ok('Do you really want to quit?')

Do you really want to quit?n

Out[261]: 0

In [262]: ask_ok('OK to overwrite the file?', 2, 'Answer correct please!')

OK to overwrite the file?asd
Answer correct please!
OK to overwrite the file?asdas
Giving up
```

27 Nyckelordsargument

```
In [136]: def parrot(voltage, state='a stiff', action='voom', type='Norwegian Blue'):
              print("-- This parrot wouldn't", action)
              print("if you put", voltage, "Volts through it.")
              print("-- Lovely plumage, the", type)
              print("-- It's", state, "!")

In [137]: parrot(1000)

-- This parrot wouldn't voom
if you put 1000 Volts through it.
-- Lovely plumage, the Norwegian Blue
-- It's a stiff !

In [138]: parrot(action = 'VOOOOOM', voltage = 1000000)

-- This parrot wouldn't VOOOOOM
if you put 1000000 Volts through it.
-- Lovely plumage, the Norwegian Blue
-- It's a stiff !

In [139]: parrot('a thousand', state = 'pushing up the daisies')
```

```
-- This parrot wouldn't voom
if you put a thousand Volts through it.
-- Lovely plumage, the Norwegian Blue
-- It's pushing up the daisies !
```

```
In [140]: parrot('a million', 'bereft of life', 'jump')
```

```
-- This parrot wouldn't jump
if you put a million Volts through it.
-- Lovely plumage, the Norwegian Blue
-- It's bereft of life !
```

```
In [141]: parrot() # obligatoriskt argument saknas
```

```
        ---------------------------------------------------------------------------

        TypeError                                 Traceback (most recent call last)

        <ipython-input-141-4b92cc05c199> in <module>()
    ----> 1 parrot() # obligatoriskt argument saknas


        TypeError: parrot() missing 1 required positional argument: 'voltage'
```

```
In [142]: parrot(voltage=5.0, 'dead') # Icke nyckelordsargument
```

```
          File "<ipython-input-142-2f7a27cf4fcb>", line 1
        parrot(voltage=5.0, 'dead') # Icke nyckelordsargument
                            ^
    SyntaxError: positional argument follows keyword argument
```

```
In [143]: parrot(110, voltage=220)
```

```
        ---------------------------------------------------------------------------

        TypeError                                 Traceback (most recent call last)

        <ipython-input-143-9bcd7eaed44e> in <module>()
    ----> 1 parrot(110, voltage=220)


        TypeError: parrot() got multiple values for argument 'voltage'
```

```
In [144]: parrot(actor='John Cleese')
```

```
        ---------------------------------------------------------------------------

        TypeError                                 Traceback (most recent call last)
```

```
    <ipython-input-144-8d2ac8bc1850> in <module>()
----> 1 parrot(actor='John Cleese')


TypeError: parrot() got an unexpected keyword argument 'actor'
```

28 Datastrukturer

28.1 Mer om listor

- append(x) - Lägger till x i slutet av listan
- extend(L) - Lägg till alla element i listan L sist
- remove(x) - Tar bort först förekomsten av x i listan
- pop([i]) - Returnerar och tar bort sista elementet i listan
- index(x) - Returnerar x index i lista
- count(x) - Returnerar antalet förekomster av x i listan
- sort() - sorterar listan
- reverse() - motsatsen

```
In [145]: a = [66.6, 333, 333, 1, 1234.5]

In [146]: print(a.count(333), a.count(66.6), a.count('x'))

2 1 0

In [147]: a.insert(2, -1)

In [148]: a.append(333)

In [149]: a

Out[149]: [66.6, 333, -1, 333, 1, 1234.5, 333]

In [150]: a.remove(333)

In [151]: a

Out[151]: [66.6, -1, 333, 1, 1234.5, 333]

In [152]: a.reverse()

In [153]: a

Out[153]: [333, 1234.5, 1, 333, -1, 66.6]

In [154]: a.sort()

In [155]: a

Out[155]: [-1, 1, 66.6, 333, 333, 1234.5]
```

28.2 del-funktionen

```
In [156]: a = [-1, 1, 66.6, 333, 333, 1234.5]

In [157]: del a[0]

In [158]: a
Out[158]: [1, 66.6, 333, 333, 1234.5]

In [159]: del a[2:4]

In [160]: a
Out[160]: [1, 66.6, 1234.5]

In [161]: del a

In [162]: a
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)

<ipython-input-162-60b725f10c9c> in <module>()
----> 1 a


NameError: name 'a' is not defined
```

28.3 Index eller dictionaries

- Associativ datastruktur
- Indexerad med nycklar
- nyckel - värde par
- Snabb access till nycklar/värden

```
In [163]: tel = {'jack': 4098, 'sape': 4139}

In [164]: tel['guido'] = 4127

In [165]: tel
Out[165]: {'guido': 4127, 'jack': 4098, 'sape': 4139}

In [166]: tel['jack']
Out[166]: 4098

In [167]: del tel['sape']

In [168]: tel['irv'] = 4127

In [169]: tel
Out[169]: {'guido': 4127, 'irv': 4127, 'jack': 4098}

In [170]: tel.keys()
Out[170]: dict_keys(['guido', 'irv', 'jack'])

In [171]: 'irv' in tel
Out[171]: True
```

28.4 Loop-tekniker för dictionaries

```
In [172]: knights = {'gallahad': 'the pure', 'robin': 'the brave'}
          knights.items()

Out[172]: dict_items([('gallahad', 'the pure'), ('robin', 'the brave')])

In [173]: for k, v in knights.items():
              print(k, v)

gallahad the pure
robin the brave

In [174]: for i, v in enumerate(['tic', 'tac', 'toe']):
              print(i, v)

0 tic
1 tac
2 toe

In [175]: for key in knights.keys():
              print(key, knights[key])

gallahad the pure
robin the brave
```

29 In- och utdatahantering

29.1 Formaterad utskrift

```
In [176]: for x in range(1, 11):
              print('%02d %03d %04d' % (x, x*x, x*x*x))

01 001 0001
02 004 0008
03 009 0027
04 016 0064
05 025 0125
06 036 0216
07 049 0343
08 064 0512
09 081 0729
10 100 1000
```

29.2 Formatkoder

```
%d      : integer
%5d     : integer in a field of width 5 chars
%-5d        : integer in a field of width 5 chars, but adjusted to the left
%05d        : integer in a field of width 5 chars, padded with zeroes from
          the left
%g      : float variable in %f or %g notation
%e      : float variable in scientific notation
%11.3e  : float variable in scientific notation, with 3 decimals, field
          of width 11 chars
%5.1f   : float variable in fixed decimal notation, with one decimal,
```

```
          field of width 5 chars
%.3f        : float variable in fixed decimal form, with three decimals,
          field of min. width
%s      : string
%-20s   : string in a field of width 20 chars, and adjusted to the left
```

29.3 Utskrift fortsättning

Önskad utdata:

```
Hello, world! sin(3.4)=-0.255541102027
```

Sammanfogning av strängar

```
In [177]: from math import *
          r = 3.4

In [178]: s = sin(r)

In [179]: print("Hello, World! sin(" + str(r) + ")=" + str(s))

Hello, World! sin(3.4)=-0.2555411020268312
```

printf-satser

```
In [180]: print("Hello, World! sin(%g)=%g" % (r,s))

Hello, World! sin(3.4)=-0.255541
```

Variabelinterpolation:

```
In [181]: print("Hello, World! sin(%(r)g)=%(s)g" % vars())

Hello, World! sin(3.4)=-0.255541
```

29.4 Filhantering

29.4.1 Öppna fil för läsning

```
In [182]: ifile = open("testfile.txt", "r")

In [183]: print(ifile)

<_io.TextIOWrapper name='testfile.txt' mode='r' encoding='UTF-8'>

In [184]: ifile.close()

In [185]: print(ifile)

<_io.TextIOWrapper name='testfile.txt' mode='r' encoding='UTF-8'>
```

29.4.2 Läsa in hela filen

```
In [186]: ifile = open("testfile.txt", "r")
          content = ifile.read()
          ifile.close()
          print(content)

This is a test file
This is the second line
This is the third line
```

29.4.3 Läsa in en rad

```
In [187]: ifile = open("testfile.txt", "r")

In [188]: row1 = ifile.readline()

In [189]: print(row1)

This is a test file

In [190]: row2 = ifile.readline()

In [191]: print(row2)

This is the second line

In [192]: row3 = ifile.readline()

In [193]: print(row3)

This is the third line

In [194]: row4 = ifile.readline()

In [195]: print(row4)


In [196]: print(len(row4))

0

In [197]: ifile.close()
```

29.4.4 Loop för att läsa in alla rader i en fil

```
In [198]: ifile = open("testfile.txt", "r")
          line = ifile.readline()
          while line!="":
              print(line)
              line = ifile.readline()
          ifile.close()

This is a test file

This is the second line

This is the third line
```

Alternativt:

```
In [199]: ifile = open("testfile.txt", "r")
          for line in ifile:
              print(line)
          ifile.close()

This is a test file

This is the second line

This is the third line
```

Mellanrummen mellan raderna beror på att vi även läser in det speciall radsluts tecknet. Vi kan komma runt detta genom att använda strängmetoden strip() enligt:

```
In [200]: ifile = open("testfile.txt", "r")
          for line in ifile:
              print(line.strip())
          ifile.close()
```

```
This is a test file
This is the second line
This is the third line
```

29.4.5 Läsa in alla rader till en lista

```
In [201]: ifile = open("testfile.txt", "r")
          lines = []
          for line in ifile:
              lines.append(line)
          ifile.close()
          print(lines)
```

```
['This is a test file\n', 'This is the second line\n', 'This is the third line\n']
```

Det finns en enklare metod för detta också:

```
In [202]: ifile = open("testfile.txt", "r")
          lines = ifile.readlines()
          print(lines)
```

```
['This is a test file\n', 'This is the second line\n', 'This is the third line\n']
```

29.4.6 Konvertering av värden till flyttal

Vi har följande fil, numbers.txt:

```
In [203]: %cat numbers.txt
```

```
1 2 3 4 5 6
7 8 9
10 11 12 13
```

Genom lite trick och stränghantering kan vi konvertera siffrorna i denna fil till tal genom följande kod:

```
In [204]: ifile = open("numbers.txt", "r")
          for line in ifile:
              parts = line.split()

              numbers = []
              for p in parts:
                  numbers.append(float(p))

              #numbers = [float(p) for p in parts]

              print(numbers)
          ifile.close()
```

```
[1.0, 2.0, 3.0, 4.0, 5.0, 6.0]
[7.0, 8.0, 9.0]
[10.0, 11.0, 12.0, 13.0]
```

29.4.7 Skriva till filer

Om vi vill skriva till filer används samma princip som för läsning. Filen öppnas först med open() och attributet "w" för skrivning. Följande exempel visar hur detta fungerar:

```
In [205]: outfile = open("outfile.txt", "w")

          for i in range(10):
              outfile.write("%d\n" % i)

          outfile.close()
```

```
In [206]: %cat outfile.txt
```

```
0
1
2
3
4
5
6
7
8
9
```