Mittuniversitetet
MID SWEDEN UNIVERSITY

DT094G – Funktioner och moduler i Python

Lennart Franked

2022-10-01

Institutionen för Data- och Elektroteknik (DET)
Mittuniversitetet

Mittuniversitetet
MID SWEDEN UNIVERSITY

Innehållsförteckning

Mittuniversitetet
MID SWEDEN UNIVERSITY

Läsanvisningar

Denna sektion behandlar [1, Kap 4.6, 4.7], läs relevanta delar och studera exempelkod.

Funktioner

- Ibland vill vi kunna återanvända kod eller strukturera upp den för läsbarhet
- Ett sätt att uppnå detta är med hjälp utav funktioner
- funktioner skapas genom att vi definierar upp önskad funktion med `def funktionsnamn():`

Listing 1: Funktion

```
1 >>> def my_function():
2 ...     print('running code in a function')
3 ...
4 >>>my_function()
5 running code in a function
```

Funktioner med indata

- Vi kan också skicka in indata i våra definierade funktioner genom att ange detta i vår funktionsdefinition.

Listing 2: Funktion med indata

```
>>> def count_letters(in_parameter):
...     number_of_letters = 0
...     for letter in in_parameter:
...         number_of_letters += 1
...     return number_of_letters
...
>>> print(count_letters('kycklingmacka'))
13
```

Mittuniversitetet
MID SWEDEN UNIVERSITY

Standardvärden på indata

- Ibland vill vi inte använda indata i en funktion.
- Ibland vill vi ha ett standardvärde på indatat, om inget annat anges.
- Detta anger vi då i vår definition av funktionen.

Exempel indata

Listing 3: Frivillig indata

```
1  >>> def count_letters(in_parameter=None):
2  ...     if in_parameter is None:
3  ...         return 'Ingen indata angavs'
4  ...
5  ...     else:
6  ...         number_of_letters = 0
7  ...         for letter in in_parameter:
8  ...             number_of_letters += 1
9  ...
10 ...         return number_of_letters
11 ...
12 >>> print(count_letters())
13 Ingen indata angavs
14 >>> print(count_letters('kycklingmacka'))
15 13
```

Exempel indata

Listing 4: Multipla indata

```
>>> def count_letters(in_parameter1, in_parameter2='kycklingmacka',
in_parameter3=None):
...     number_of_letters = 0
...     if in_parameter3 is None:
...         wordlist = in_parameter1+in_parameter2
...     else:
...         wordlist = in_parameter1+in_parameter2+in_parameter3
...
...     for letter in wordlist:
...         number_of_letters += 1
...
...     return number_of_letters
...
>>> print(count_letters('juice'))
18
>>> print(count_letters(in_parameter1='juice',in_parameter3='kycklingmacka'))
31
```

Okänt antal indata

- Ibland vet vi inte hur många variabler vi vill ha med i en funktion.
- vi kan då ange `*args` som inparameter till vår funktion.
- indatat behandlas då som en tupel.

Exempel läsa in okänt antal indata

Listing 5: okänt antal indata

```
>>> def unknown_input(*args):
...     for arg in args:
...         print(type(arg), arg)
...
>>> unknown_input('foo', 'bar', 42, ['sundsvall','miun', 'ist'])
<class 'str'> foo
<class 'str'> bar
<class 'int'> 42
<class 'list'> ['Sundsvall', 'miun', 'ist']
>>>
```

Mittuniversitetet
MID SWEDEN UNIVERSITY

Lambdafunktioner

- Anonyma funktioner eller lamdafunktioner fungerar precis som en vanlig funktion, dock begränsad till ett endast ett uttryck.
- Finns inget användningsområde för lambda som man inte kan lösa med funktion, utan finns mest för läsbarhet och bekvämlighet.

Exempel lambda

Listing 6: Lambda

```
>>> square = lambda x : x**2
>>> cube = lambda x : x**3
>>>
>>> print(square(5))
25
>>> print(cube(5))
125
```

Listing 7: klassisk funktion

```
>>> def square(x):
...     return x**2
>>> def cube(x):
...     return x**3
...
>>> print(square(4))
16
>>> print(cube(4))
64
```

Innehållsförteckning

Mittuniversitetet
MID SWEDEN UNIVERSITY

Läsanvisningar

Denna sektion behandlar [1, kapitel 6], läs relevanta delar och studera exempelkod.

Mittuniversitetet
MID SWEDEN UNIVERSITY

Moduler i Python

- När vi skapar en fil innehållandes python-kod så kallas denna fil för modul.
- Oftast namnger man dessa filer med ändelsen ‘.py’
- Varje modul ges ett namn, baserat på dess filnamn.
- Detta namn lagras som en global variabel `__name__`

Modulrelaterade globala variabler

Listing 8: __name__

```
1 >>> print(__name__)
2 __main__
3 >>>
```

Importera kod från modul

Listing 9: exponentiation.py

```
1  def square(x):
2      return x**2
3
4  def cube(x):
5      return x**3
```

Listing 10: Pythonskalet

```
1  >>> import exponentiation
2  >>> print(exponentiation.square(4))
3  16
4  >>> print(exponentiation.cube(4))
5  64
```

Importera kod från modul forts.

Listing 11: exponentiation.py

```
1  def square(x):
2      return x**2
3
4  def cube(x):
5      return x**3
```

Listing 12: Pythonskalet

```
1  >>> from exponentiation import square, cube
2  >>> #alternativt
3  >>> from exponentiation import *
4  >>>
5  >>> print(square(4))
6  16
7  >>> print(cube(4))
8  64
```

Importera kod från moduler forts.

- Då man importerar med hjälp av wildcard läser python-tolken in allt som inte börjar med _
- Man kan lista allt som går att importera från en modul med `dir`

Listing 13: Pythonskalet

```
>>> import exponentiation
>>> print(dir(exponentiation))
['__builtins__', '__cached__', '__doc__', '__file__', '__loader__', '__name__',
 '__package__', '__spec__', 'cube', 'square']
>>>
```

Exekvera moduler som skript

- Vill man exekvera en modul som ett skript lägger man in ett vilkor.

Listing 14: Styra start av modul

```
1  def square(x):
2      return x**2
3
4  if __name__ == '__main__':
5      print(square(16))
```

Sökvägar för moduler

- Då man utför en import så söker python-tolken efter moduler i följande ordning.
  1. Den sökväg där skriptet som körs finns.
  2. PYTHONPATH
  3. Default standard sökväg för den specifika installationen.

Paket

- Man kan gruppera ihop sina moduler i paket, genom att lägga dem i en gemensam katalog.
- Paketet ‘skapas’ genom att man lägger en __init__.py fil i mappen.
- Man får åtkomst till specifika moduler i ett paket genom att ange sökvägen i punkt-form

```
1 import mapp.modul.funktion
```

Referenser I

[1] *The Python Tutorial*. 2016. URL: https://docs.python.org/3/tutorial/index.html.

Mittuniversitetet
MID SWEDEN UNIVERSITY