Python Introduction for Programmers

Edwin van der Helm

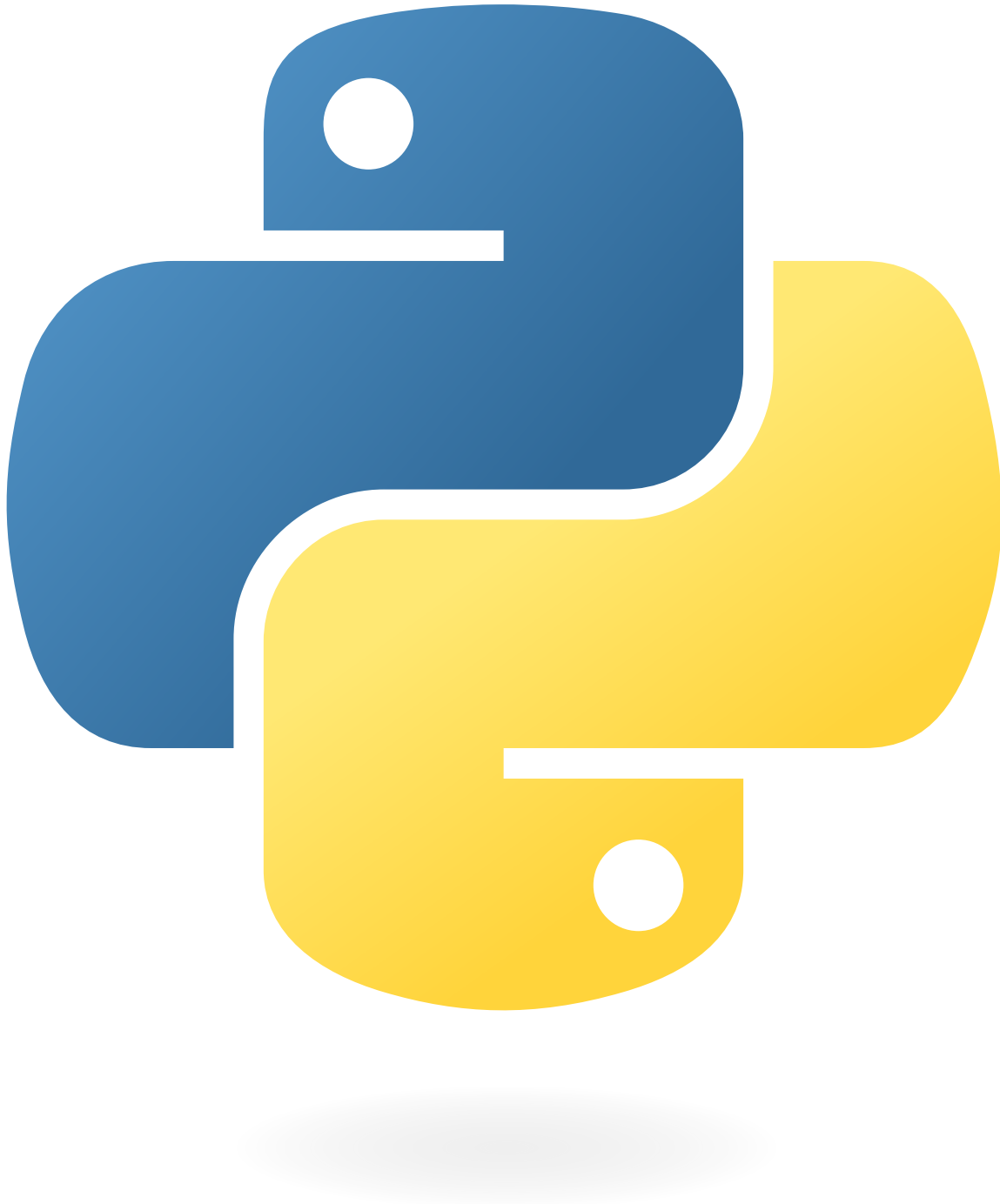
Sterrewacht Leiden

Januari 21, 2014

strw.leidenuniv.nl/∼vdhelm/teaching.php

Python Introduction for Programmers

Edwin van der Helm



Scripts

```
1 print "Hello World"
```

```
$ python
>>> print "Hello World"
Hello World

$ python hello_world.py
Hello World
```

Variables

```
1  x = 2
2  y = 2 + 2
3  z = x * 9
4
5  x = y = z = 0
6
7  x, y = 3, 4
8  y, x = x, y
9
10 x = str(x)
11
12 print "y :", y
```

Math and Numbers

```
1  print 2+3 # = 5
2  print 2-3 # = -1
3  print 10*3 # = 30
4  print 12/3 # = 4
5  print 10%3 # = 1
6  print 10**3 # = 1000
7
8  print 10 + 3 * 5   # first *,/ then +,-
9  print (10 + 3) * 5
10 print -1**2 # = -(1**2)
11 print 1/2 # = 0
12 print 1./2. # = 0.5
13 print 1.//2. # = 0.0
```

```
1  from __future__ import division
2  print 1/2
```

Math and Numbers

```
1  # Float
2  2.3
3  -4.
4  .1
5  2.99e10
6  -1e1, -1E1
7
8  # Long
9  2**1000
10 2L, 2l
11
12 # Complex
13 1J
14 2. + 3j
15 (2.+3j).real
16 (2+3j).imag
```

```
1  # Explicit
2  float(2)
3  float(2**1000)
4  int(2.3)
5  int(-2.3)
6  int(2**1000)
7  long(2)
8  str(2)
9  complex(1,2)
10
11 # Test
12 type((2-3j))
```

Strings and Lists

```
1 x = 'test string'
2
3 'doesn\'t'
4 "doesn't"
5 '"Yes," he said.'
6
7 """ This is a multi
8 line string with
9 an entire story in it."""
```

Strings and Lists

```
1  s = "I have a telescope!    "
2  len(s)
3  'tel' + "escope" # Concatination
4  'telescope' + 100 # won't work
5  'telescope' + str(100)
6  'telescope' * 100 # Be careful
7  'telescope' * '100'
8
9  s[0] # First character
10 s[1] # Second character
11 s[2:10] # Substring
12 s[100] # Error
13 s[-1] # Last character
14
15 s = s.strip()
16 s1 = s.split()
17 s2 = s.split('e')
18 s = s.replace("telescope", "computer")
19 "Name: {0}, age: {1}".format('John', 35)
```

Strings and Lists

```
1  x = ["I", "have a",  "telescope!", 4]
2
3  # Slices
4  x[0]
5  x[1:]
6  x[::2]
7  x[2] = "computer!"
8  x[1:3] = 'bla'
9
10 # Dictionary
11 y = {1323:'Bob', 6543:'Alice'}
12 print y[1323]
13 y = {'Bob':1323, 'Alice':6543}
14 print y ['Alice']
```

Strings and Lists

```
1 # List comprehensions
2 print range(10)
3 squares = [i**2 for i in range(30)]
4 dict_squares = [i:i**2 for i in range(30)]
```

Strings and Lists

```
1  x = ["I", "have a",  "telescope!", 4]
2
3  " ".join(x[:-1])
4  x.append("yes")
5  x.extend(range(6))
6  x.insert(2, "good")
7  x.remove('I')
8  x.pop()
9  x.index('have a')
10 x.count(4)
11 x.sort()
12 x.reverse()
```

Indentation

```
1  for i in range(3, 20, 2):
2      print "This is part of the loop", i
3      print "This is also part of the loop"
4  print "This is not part of the loop"
5
6  # Use 4 spaces or one tab as indentation.
7  # Please be consistent with your indentation
8
9  for i in range(5):
10     for j in range(5):
11         print "Inner loop", i, j
12     print "Outer loop", i, j
13 print "Not in the loop"
```

Indentation

```
1  for letter in 'Python':
2      if letter == 'h':
3          break
4      print 'Current Letter :', letter
5
6  for letter in 'Python':
7      if letter == 'h':
8          continue
9      print 'Current Letter :', letter
10
11 for letter in 'Python':
12     if letter == 'x':
13         break
14     print 'Current Letter :', letter
15 else:
16     print 'No x in python'
```

Logic

```
1  if 'Steven' in ['Bob', 'Steven', 'Fred']:
2      print 'Here!'
3
4  if 'Carol' not in ['Bob', 'Steven', 'Fred']:
5      print 'Away!'
6
7  test = a == b
8  if test:  print 'Equal'
9
10 x = int(raw_input("Enter integer: "))
11 if x == 0:
12     print 'Zero'
13 elif x == 1:
14     print 'One'
15 else:
16     print 'Not binary'
```

Logic

```
1  "sub" in "mysubstring"
2  "mystring".startswith("myst")
3
4  0 and "not empty"
5  False or "" or [1,2,3] or 42
6  "string" is not None
7
8  2 < x <= 10
9
10 x = y = [1,2,3]
11 [1,2,3] == x
12 [1,2,3] is x
13 y is x
```

Functions

```
1  def my_function(x, y):
2      a = x + y
3      print a
4
5  z = 10
6  my_function(z, 2)
7  my_function(z, z-2)
8
9  # Return values
10 def my_function_2(x, y):
11     a = x + y
12     b = x * y
13     return a, b
14
15 if __name__ == "__main__":
16     print my_function_2(1, 2)
17     m, p = my_function_2(1, 2)
```

Parameters

```
1  def my_function(x, y):
2      print  x + y**2
3
4  if __name__ == "__main__":
5      # Calling by keyword
6      my_function(y=2, x=3)
7
8  # default values
9  def my_function_2(x=1, y=1):
10     print x + y**2
11
12 if __name__ == "__main__":
13     my_function_2(2, 3)
14     my_function_2(y=2)
15     my_function_2()
```

Parameters

```
1  def my_function(x, y):
2      print  x + y**2
3
4  if __name__ == "__main__":
5      arguments = [2, 3]
6      my_function(*arguments)
7
8      params = {'y':3, 'x':5}
9      my_function(**params)
10
11 def my_function_2(*args, **kwargs):
12     for arg in args:
13         print arg
14
15     for name, value in kwargs.items():
16         print name, "has value", value
17
18 if __name__ == "__main__":
19     my_function_2(1, 2, 3, x=4, y=5, z=6)
```

Advanced tricks

```
1  def todo_function(x):
2      pass
3
4  def change_and_call(x, y,
5                      func=todo_function)
6      a = x % y
7      func(a)
8
9  def other_function(x):
10     def inner_function(x)
11         print x
12
13     change_and_call(x, x+1, inner_function)
14
15 if __name__ == "__main__":
16     change_and_call(1, 2, other_function)
17
18     print change_and_call(1, 2,
19                           lambda y: y**2)
```

Classes

```
1  class Planet(object):
2      def __init__(self, name, mass):
3          self.mass = mass
4          self.name = name
5
6      def display_planet(self):
7          s = "Name: {p.name} mass: {p.mass}"
8          print s.format(p=self)
9
10 if __name__ == "__main__":
11     planet1 = Planet("Earth", 5.972e24)
12     planet2 = Planet("Venus", 4.867e24)
13     planet1.display_planet()
14     planet2.display_planet()
```

Classes

```
1  class Gas_Giant(Planet):
2      def display_planet(self):
3          s="{p.name}, Gas giant, M={p.mass}"
4          print s.format(p=self)
5
6      def orig_display(self):
7          Planet.display_planet(self)
8
9  if __name__ == "__main__":
10     planet3 = Gas_Giant("Jupiter", 1.9e+27)
11     planet3.display_planet()
12     planet3.orig_display()
13
14     planet3.radius = 7.15e+07
15     print hasattr(planet1, "radius")
16
17     planet3.show = planet3.display_planet
18     planet3.show()
```

Packages

```
1  import numpy
2
3  from matplotlib import pyplot
4  import matplotlib.pyplot as plt
5  from matplotlib import pyplot, gridspec
6
7  from astropy.io import fits
8
9  # Please don't use:
10 from amuse.lab import *
```

Numpy

```
1 import numpy
2
3 a = numpy.array([1,3,5,7,9])
4 b = numpy.array([3,5,6,7,9])
5
6 print a + b
7 print a * b
8 print 2 * a
9 print a**2
```

Numpy

```
1  import numpy
2
3  a = [[i + 10*j for i in range(10)]
4         for j in range(11)]
5  a = numpy.array(a)
6  print a
7  print a.shape
8
9  # Slices
10 print a[3,4]
11 print a[:,5]
12 print a[2:-3,:3]
13 print a[2:-3,:9:2]
14
15 print 22 < a
16 print a[22 < a]
17 print (20 < a) & (a < 40)
```

Numpy

```
1  print a.astype(float)
2  b = a.copy()
3  print a is b
4
5  print a.sum()
6  print a.mean()
7  print a.std()
8  print a.max()
9  print a.min()
10
11 test = (20 < a) & (a < 40)
12 print test.all()
13 print test.any()
14 print test.sum()
```

Numpy

```
1  b = a**2
2  c = a.transpose()
3  print c
4  print c.dot(b)
5  print numpy.cross(a[:2,1], a[-2:,2])
6  print numpy.cross([1, 2, 3], [0, 1])
7
8  ma = numpy.asmatrix(a)
9  mb = numpy.asmatrix(b[:5,:5])
10 mc = numpy.asmatrix(c)
11 print ma * mc
12 print mb**2
13 print mb.T
14 print mb.I
```

Numpy

```
$ cat planets.txt
# Name, mass (kg), radius (m)
Mercury   3.3e+23     2.44e+06
Venus     4.87e+24    6.05e+06
Earth     5.97e+24    6.38e+06
Mars      6.42e+23    3.4e+06
```

```
1 import numpy as np
2 mass, radius = np.loadtxt("planets.txt",
3                           dtype=float,
4                           usecols=(1,2),
5                           unpack=True)
```

Astropy

```
1  from astropy.io import fits
2
3  # open() has many keyword options.
4  fits_file = fits.open('input.fits')
5  print fits_file.info()
6
7  print fits_file[0].header()
8
9  # The data can be read as a numpy array
10 data = fits_file[0].data
11 print data.shape
12 print data.mean()
13 print data[10:-10, 10:-10]
```

Astropy

It also includes:

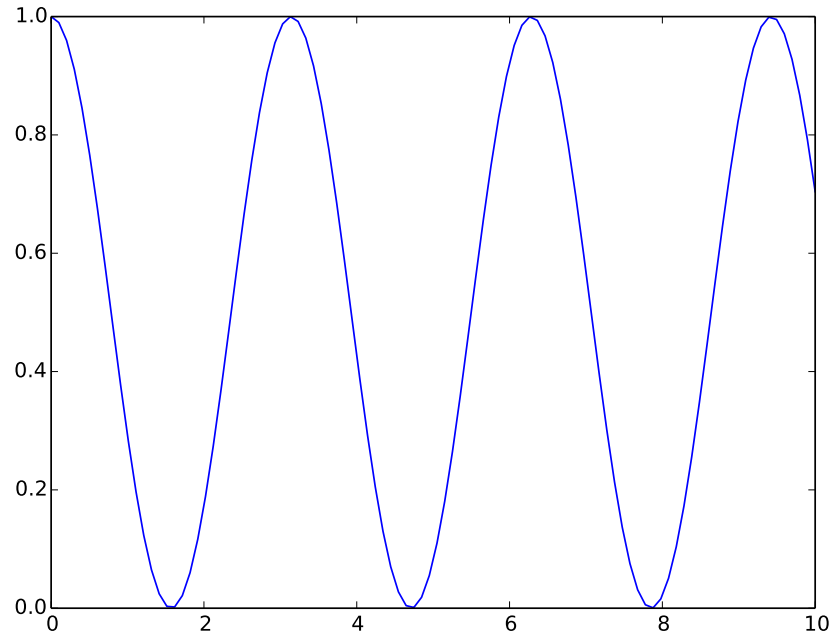
- Units and coordinates
- Fitting
- Convolution and filtering
- Cosmology
- Astrostatistics tools
- ...

PyRAF

```
1  import os
2  from pyraf import iraf
3
4  for img in os.listdir("fits_files/"):
5      iraf.imstat(img)
6      newimg = img.replace('sci', 'sig')
7      iraf.imcalc(img, newimg, 'sqrt(img)')
8
9  iraf.imstat.nclip = 3
10 iraf.imstat.lsigma = 5
11 iraf.imstat.usigma = 5
12
13 # now imstat uses sigma clipping
14 iraf.imstat(im1)
15
16 # revert defaults using unlearn
17 iraf.unlearn('imstat')
```

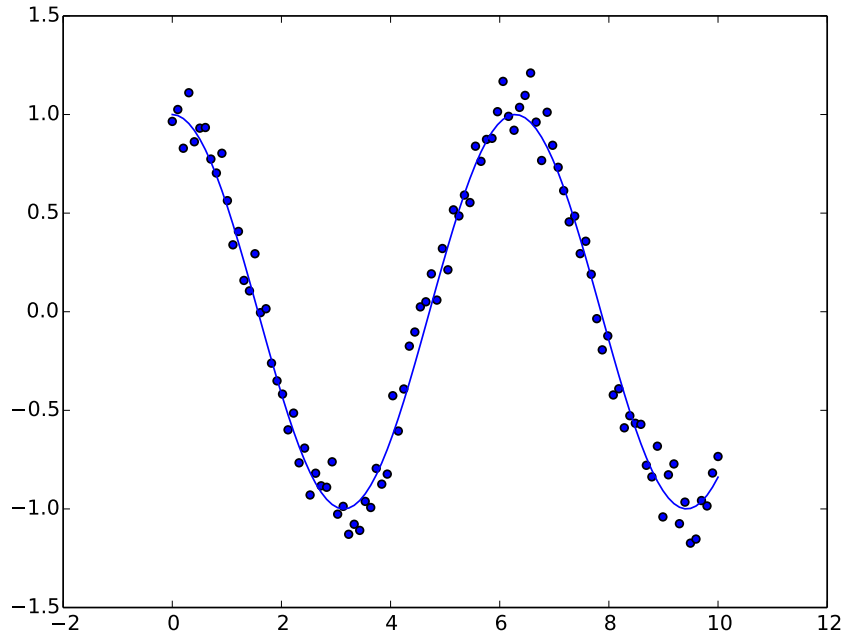
Pyplot

```
1 import numpy
2 from matplotlib import pyplot
3
4 x = numpy.linspace(0, 10, 100)
5 y = numpy.cos(x)**2
6
7 pyplot.plot(x, y)
8 pyplot.show()
```



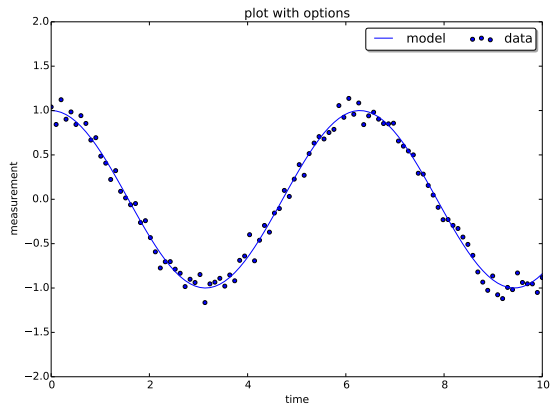
Pyplot

```
1 | x = numpy.linspace(0, 10, 100)
2 | m = numpy.cos(x)
3 | d = m + numpy.random.normal(scale=0.1,
4 |                             size=100)
5 | pyplot.plot(x, m)
6 | pyplot.scatter(x, d)
7 | pyplot.savefig("model.pdf")
```



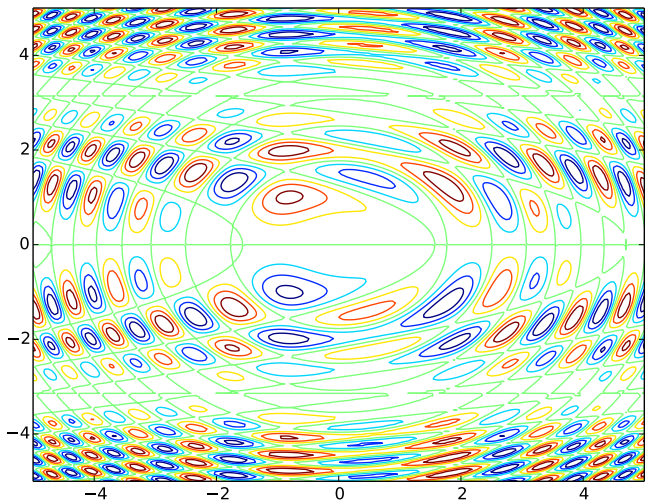
Pyplot

```
1  figure = pyplot.figure(figsize = (10, 7))
2
3  pyplot.plot(x, model, label="model")
4  pyplot.scatter(x, data, label="data")
5
6  pyplot.xlim([0, 10])
7  pyplot.ylim([-2, 2])
8  pyplot.xlabel("time")
9  pyplot.ylabel("measurement")
10 pyplot.title("plot with options")
11 pyplot.legend(loc="upper right", ncol=2,
12                 fancybox=True, shadow=True)
```



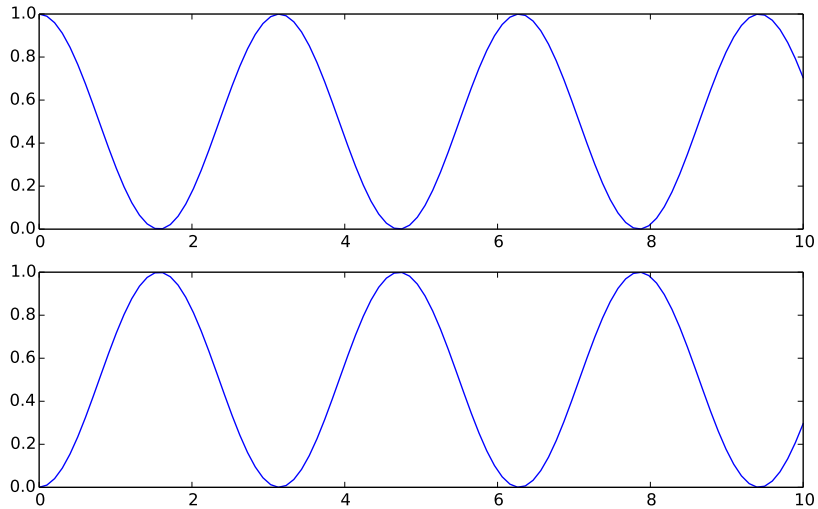
Pyplot

```
1 X = numpy.linspace(-5, 5, 200)
2 Y = numpy.linspace(-5, 5, 200)
3 X, Y = numpy.meshgrid(X, Y)
4 Z = (numpy.sin(X**2 + Y**2)
5      * numpy.cos(X + Y**2)
6      * numpy.sin(Y))
7
8 pyplot.contour(X, Y, Z)
9 pyplot.savefig("contour.pdf")
```



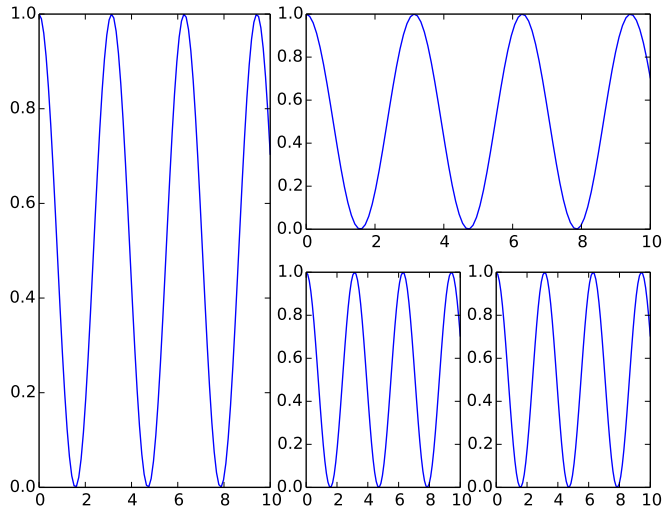
Pyplot

```
1 x = numpy.linspace(0, 10, 100)
2 y = numpy.cos(x)**2
3 y2 = numpy.sin(x)**2
4
5 figure = pyplot.figure(figsize = (10, 6))
6 subplot = figure.add_subplot(211)
7 pyplot.plot(x, y)
8 subplot = figure.add_subplot(212)
9 pyplot.plot(x, y2)
```



Pyplot

```
1  from matplotlib import pyplot, gridspec
2  gs = gridspec.GridSpec(2, 3,
3                         width_ratios=[3,2,2])
4  pyplot.subplot(gs[:, 0])
5  pyplot.plot(x,y)
6  pyplot.subplot(gs[0, 1:])
7  pyplot.plot(x,y)
8  pyplot.subplot(gs[1, 1])
9  pyplot.plot(x,y)
10 pyplot.subplot(gs[1, 2])
11 pyplot.plot(x,y)
```



Amuse

```
1  from amuse.lab import *
2
3  number_of_stars = 100
4  size = 2 | units.pc
5
6  masses = new_salpeter_mass_distribution(
7      number_of_stars, mass_min=2|units.MSun)
8
9  converter = nbody_system.nbody_to_si(
10     masses.sum(), size)
11 stars = new_plummer_model(
12     number_of_stars,
13     convert_nbody=converter)
14
15 stars.mass = masses
```

Amuse

```
1  gravity = Hermite(converter)
2  stellar = SeBa()
3
4  gravity.particles.add_particles(stars)
5  stellar.particles.add_particles(stars)
6
7  interaction = (stellar.particles.
8                 new_channel_to(
9                 gravity.particles,
10                attributes=["mass", "radius"]))
```

Amuse

```
1 time = 0 | units.Myr
2 while time <= end_time:
3     interaction.copy()
4
5     time += timestep
6     gravity.evolve_model(time)
7     stellar.evolve_model(time)
8
9     plot_or_save(gravity, stellar, time)
```

Help

- http://www.strw.leidenuniv.nl/python
- google

```
1 import numpy
2 print numpy
3 print dir(numpy)
4 print [s for s in dir(numpy) if "min" in s]
```

```
>>> import numpy
>>> help(numpy)
>>> help(numpy.linspace)
```

```
$ python -m pdb my_program.py
```