Python: modules and namespaces

- how to expand and organize Python functions
- **modules** collect related functions, constants etc.

```
>>> from numpy import *
>>> sqrt(1.0)
```

- **namespaces** disambiguate duplicate names

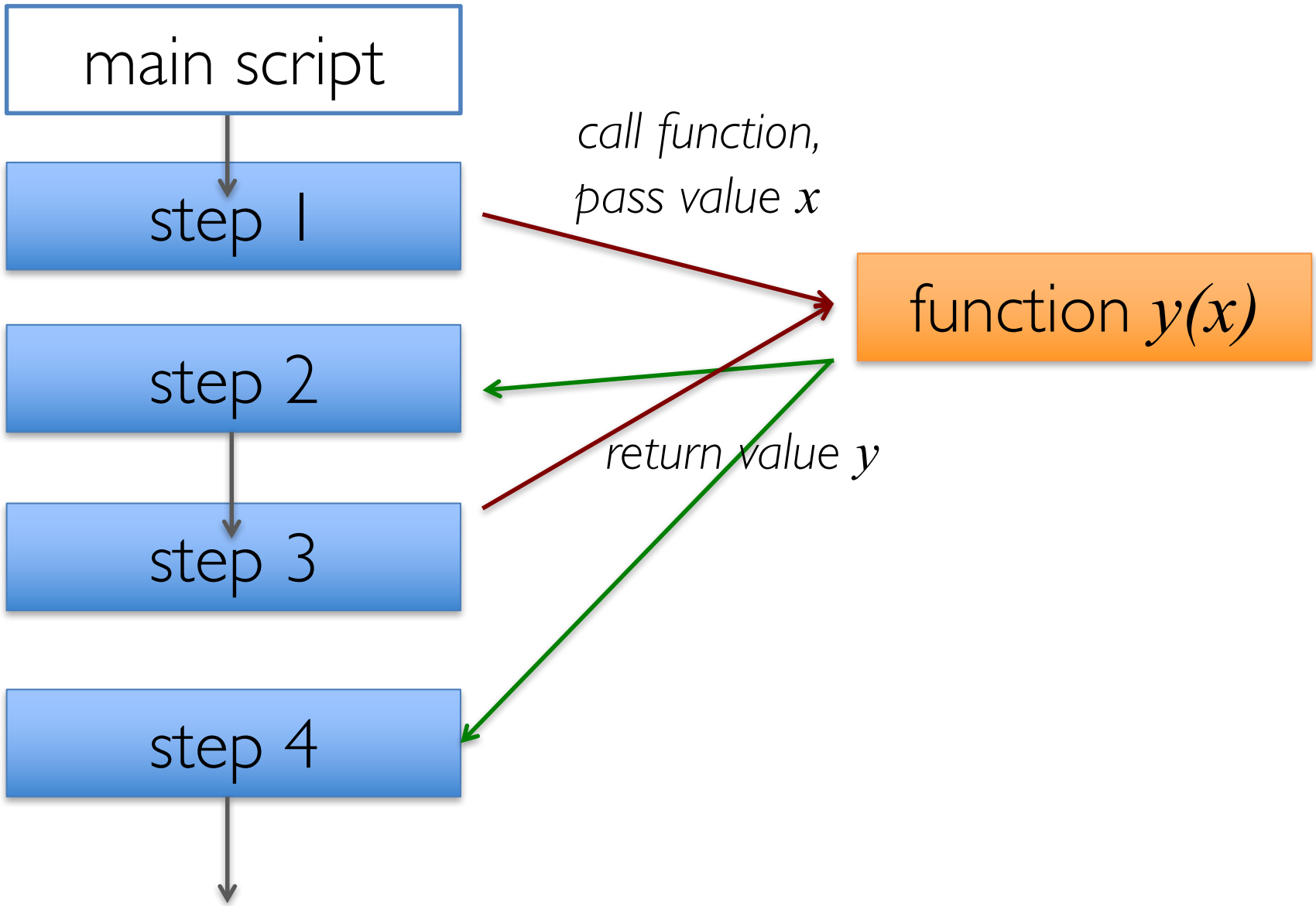
  (like area codes to phone numbers: use if you roam!)

```
>>> import numpy
>>> import math
>>> numpy.sqrt(1.0)
>>> math.sqrt(1.0)
>>> import numpy as np
>>> np.sqrt(1.0)
>>> from pylab import *
```

(NumPy + part of Matplotlib)

Functions, $y(x)$

- building blocks for modularized programming and code re-use ("Don't Repeat Yourself")



Python functions

- building blocks for modularized programming:
  - *built-in* or *user-defined* (***best to place at top of script***)
- syntax *(note critical use of indents)* :

```
def function_name(input_arguments) :
    """Documentation of purpose and use."""
    body of function
    return output_values
```

- example to calculate area of a square :

```
def area_square(side) :
    """Calculate the area of a square."""
    area = side**2
    return area
```

- execute by typing command name with input(s):

```
>>> area = area_square(11.33)
```

Python: function syntax example

- function definition:

```
def ATM(pin_number) :
    """ input pin number and get cash out"""
    cash_out = pin_number * 2.0
    return cash_out
```

- poor usage (generally):

```
>>> ATM(1234)   # you don't want to keep the cash?
>>> ATM=1234    # you just redefined the machine!
```

- good usage:

```
>>> cash1 = ATM(1234)
>>>
>>> pin2 = 567
>>> cash2 = ATM(pin2)
```

Revise planet-volume script with a function

```
# volume_planet2.py:
#    calculate volume of a planet
from pylab import *

def planet_volume(radius) : # define function
  ''' Calculate the volume.'''
  volume = (4.0/3.0) * pi * radius**3
  return volume

r = eval(input('Enter the radius: '))
v = planet_volume(r)      # use the function
print('The volume is %.3g km^3.' % (v))
```

More Python function rules

- functions must be defined before they are called
  - *best to put all functions at beginning of script*
- functions normally accept input variable(s) and return output variable(s) *but do not have to:*

```
def test_func():
    print('Hello world')
```

- functions do **not** have to do math *(see above)*
- provide variable to store output:

```
>>> var_out = sample_function(var_in)
```

- functions use internal, "quarantined" variables

Functions with multiple inputs

```
# calculate mass of a planet
```

$$M(\rho, r) = \frac{4\pi}{3}\rho r^3$$

```
def planet_mass(r, rho) : # define the function
    ''' Calculate the volume.'''
    mass = (4./3.)*pi * rho * r**3
    return mass

r,rho = eval(input('Enter the radius and density:'))
M = calc_mass(r,rho)      # use the function
print('The mass is %.1e kg.' % (M))
```

Computer programming and good practices

- organize, structure (modularize), document, debug
- clarity trumps cleverness
- universal characteristics:
  constants, variables, data types, flow control
- use variable names that are intuitive
  (‘`r`’ for radius, ‘`t`’ for time, etc.)