

Importing Libraries

Libraries are collections of *functions* that are useful.

```
>>> import math
>>> math.sin(1)
0.8414709848078965
```

Getting Help

dir and **help** or **.__doc__** are your friends

dir shows the "methods" (or *functions*) on objects that you can call.

```
>>> dir(math)
['__doc__', '__file__', '__name__', '__package__', 'acos',
'acosh', 'asin', 'asinh', 'atan', 'atan2', 'atanh', 'ceil',
'copy', 'cosh', 'cos', 'coshf', 'degrees', 'e', 'exp', 'fabs',
'factorial', 'floor', 'fmod', 'frexp', 'fsum', 'hypot',
'isinf', 'isnan', 'ldexp', 'log', 'log10', 'log1p', 'log2',
'log10f', 'log2f', 'modf', 'pi', 'pow', 'radians', 'sin', 'sinh',
'sqrt', 'tan', 'tanh', 'trunc']
```

help shows the help documentation for a "method".

```
>>> help(math.sin)
Help on built-in function sin in module math:
<BLANKLINE>
sin(.x)
  sin(x)
  <BLANKLINE>
  the sine of x (measured in radians).
  <BLANKLINE>
```

A simple program

Type the following into a file named **name.py** using a "text editor":

```
def greeting():
    your_name = raw_input('Please type your name:')
    if your_name == 'Matt':
        print "Hi Matt!"
    else:
        print name

greeting()
```

Run your program by typing this into a *terminal*: **python name.py**

Credits

Matt Harrison

©2010, licensed under a Creative Commons Attribution/Share-Alike (BY-SA) license.

Elementary Python

Starting Python

This is for Python version 2. Windows users will need to install Python from www.python.org. It is included in Mac and Linux computers.

Type **python** into a *terminal*. (Windows => run... -> cmd -> python) (Mac Applications -> Terminal -> python)

All the examples here contain an *terminal* prompt (**>>>**). Don't type in the prompt.

Variables

Variables are chunks of memory the computer uses to store information. *Variables* can have different *types*.

Strings

Strings hold character data, like words and sentences.

```
>>> name = 'Matt'
>>> lastname = 'Harrison'
>>> about = .....Matt was born many
... moons ago.....He likes
... platters.....
```

Strings can be "concatenated" or joined.

```
>>> full_name = name.capitalize() + ' ' + lastname
>>> print full_name
Matt Harrison
```

Numbers

Integers are whole numbers

```
>>> age = 10
>>> age_last_year = age - 1
```

Floats are real numbers

```
>>> miles = 10.0
>>> gallons = 3.2
```

Standard math operators are allowed (**+**, **-**, ***** (multiplication), **/** (division), ****** (power))

```
>>> miles_per_gallon = miles / gallons
>>> print miles_per_gallon
3.125
```

Converting variables

Turn a string into a number (use **int** or **float**). Notice that **pets** is really a *string*.

```
>>> pets = '4'
>>> num_pets = int(pets)
```

Turn a number into a string

```
>>> pets_2 = str(num_pets)
```

Lists

Lists are used to hold groups of data.

```
>>> names = ['name', 'fred', 'george']
>>> names.append('charles')
range can be used to create a list of numbers.
>>> range(2, 10) # numbers from 2 up to 10
[2, 3, 4, 5, 6, 7, 8, 9]
```

Comments

comments are used as reminders to programmers. Computers ignore *comments*, but they are useful to humans. Use **#** to start *comments*

```
>>> grade = 4 # need to keep track of grade
>>> # You can also have just a comment by itself
```

Output

Use **print** to write to screen

```
>>> print name
Matt
>>> print miles_per_gallon
3.125
```

Input

raw_input allows you to type data in. Here it will be stored in **friend**. You might need to convert your *variable* into a number; since the input is returned as a *string*.

```
>>> friend = raw_input("Enter a friend's name")
>>> age = int(raw_input("Enter a friend's age"))
```

Functions

Functions are reusable code chunks.

```
>>> def add_5(number):
...     return number + 5
>>> add_5(2)
7
```

Whitespace

Python denotes *blocks* by indentation. Note that in the *function* above, the line "return number + 5" was indented. *Blocks* must:

- Be indented consistently (4 spaces is ok)
- Be preceded by a ":"

Conditionals

Sometimes you want to take action if a statement is "truthy".

"Truthy"	"Falsy"
'not empty'	''
1	0
3	
[1,2,3]	[]
True	False

Common Conditionals

Syntax	Meaning
>	Greater than
<	Less than
>=	Greater than or equal
<=	Less than or equal
==	Equal to
!=	Not equal to

If statements

if statements indent the "block" following the truth. There can be zero or more **elif** statements and one or zero **else** statements.

```
>>> grade = 95
>>> if grade > 90:
...     print "A"
... elif grade > 80:
...     print "B"
... else:
...     print "C"
A
```

Looping

Sometimes you want to repeat logic. **while** or **for** statements allow that.

```
>>> num = 2
>>> while num > 0:
...     print num
...     num = num - 1
2
1
>>> for num in range(1, 3):
...     print num
1
2
```

Can **break** out of loops

```
>>> for num in range(100):
...     print num
...     if num == 1:
...         break
0
1
```