

Lab 5 – Importing

```
import  
random Module  
math Module
```

Python’s use of modules is a way to break the solution to a problem down into meaningful chunks.

There are dozens of standard Python modules that solve dozens of problems for us.

We’re not ready to look at modules in any depth, that comes later.

This lab has just a couple of steps to start using modules so that you can make use of two very simple modules: **math** and **random**.

A Python module extends the Python language by adding new classes of objects (we will learn about classes and objects toward the end of this course), new functions and helpful constants (variables).

The **import** statement tells Python to get a module. This module is then added to our working environment.

Donald Keidel, Ph.D. 2016

We will use the simplest form: **import**.

import <module name>

This statement will tell Python to locate the module and provide us with the definitions in that module.

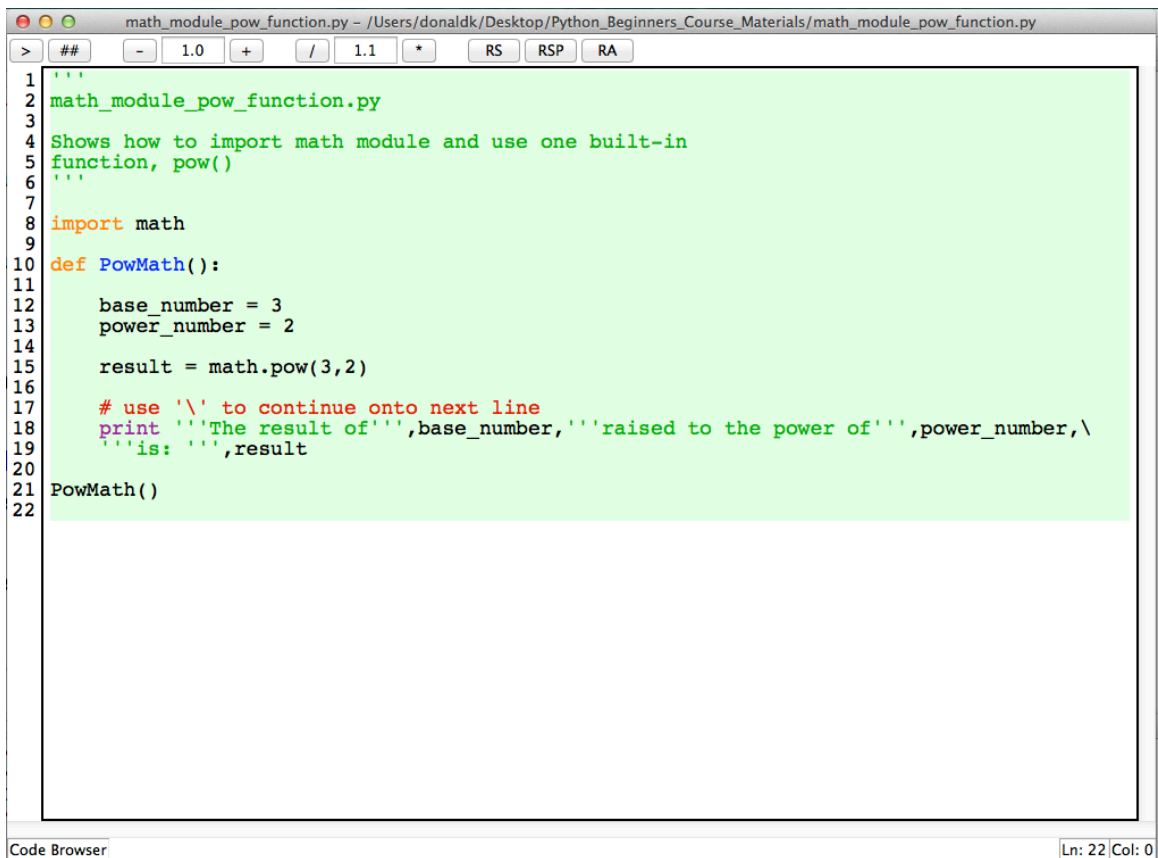
Only the name of the module is added to the local names that we can use.

When we import module **math**, we get a cosine function that we refer to with “module name dot function name”

For example, `math.cos()`

You only *need* to import a module once to tell Python you will be using it; meaning each time you run the Python program.

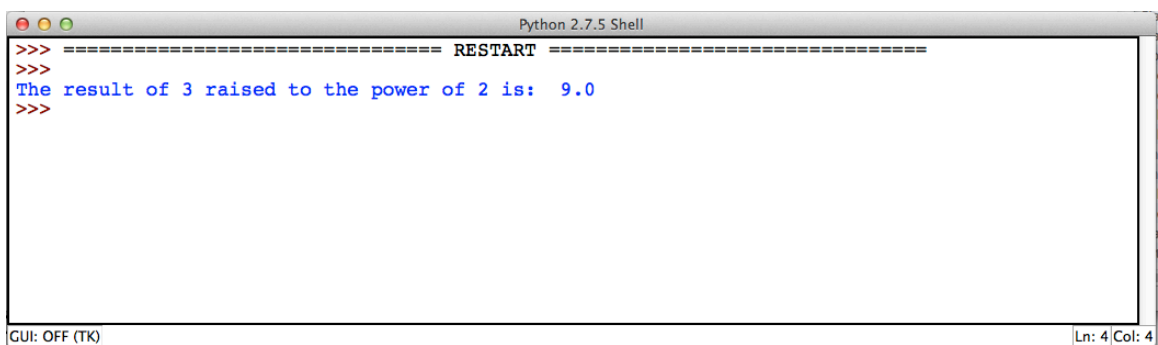
Each time you exit from the Python program (or turn your computer off, which exits all your programs), everything is forgotten. Next time you run the Python program, you’ll need to provide the **import** statements to add the modules to Python for your current session.



```
1 '''
2 math_module_pow_function.py
3
4 Shows how to import math module and use one built-in
5 function, pow()
6 '''
7
8 import math
9
10 def PowMath():
11     base_number = 3
12     power_number = 2
13
14     result = math.pow(3,2)
15
16     # use '\' to continue onto next line
17     print 'The result of',base_number,'raised to the power of',power_number,\
18           'is: ',result
19
20 PowMath()
21
22
```

Code Browser Ln: 22 Col: 0

Donald Keidel, Ph.D. 2016



```
>>> ===== RESTART =====
>>>
>>> The result of 3 raised to the power of 2 is: 9.0
>>>
```

GUI: OFF (TK) Ln: 4 Col: 4

```
random_module_randrange_function.py - /Users/donaldkeidel/Desktop/Python_Beginners_C...
1  """
2  random_module_randrange_function.py
3
4  The random module defines functions that simulate random events.
5  This includes coin tosses, die rolls and the spins of a Roulette wheel.
6
7  This program will toss two coins and when both are the same type, will
8  quit.
9  """
10
11 import random
12
13 def GetSameType():
14     toss_counter = 1
15
16     while True:
17         if toss_counter == 1:
18             print toss_counter, 'time tossing the coins'
19         if TossCoins() == True:
20             print 'Both coins are of the same type.'
21             break
22         else:
23             toss_counter = toss_counter + 1
24             print toss_counter, 'times tossing the coins'
25
26 def TossCoins():
27     """Uses random.randrange(1, 2) to emulate the flip of a coin.
28     """
29
30     coin_one = ConvertToType(random.randrange(1, 3))
31     coin_two = ConvertToType(random.randrange(1, 3))
32
33     return CheckSame(coin_one, coin_two)
34
35 def ConvertToType(coin_num_rep):
36
37     if coin_num_rep == 1:
38         return 'Heads'
39     elif coin_num_rep == 2:
40         return 'Tails'
41
42 def CheckSame(coin_one, coin_two):
43
44     if coin_one == coin_two:
45         return True
46     else:
47         return False
48
49 GetSameType()
50
51
52
```

Code Browser

Ln: 19 Col: 55

```
IPython Shell
Python 2.7.6 (default, Sep  9 2014, 15:04:36)
[GCC 4.2.1 Compatible Apple LLVM 6.0 (clang-600.0.39)] on darwin

Welcome to the IdleX IPython shell.

Python 2.7.6 (default, Sep  9 2014, 15:04:36)
Type "copyright", "credits" or "license" for more information.

IPython 2.4.1 -- An enhanced Interactive Python.
? -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help -> Python's own help system.
object? -> Details about 'object', use 'object??' for extra details.

In [1]:
===== RESTART =====

In [1]:
1 time tossing the coins
2 times tossing the coins
Both coins are of the same type.

In [1]: |

GUI: OFF (TK)
Ln: 25 Col: 8
```

Lab 5 – Exercises:

1. Open the Python prompt on the commandline and type the following:

```
>>> help('modules')
```

If the module was written correctly, you can determine how to use the module by doing the following:

Import the module to allow you to use it

```
>>> import subprocess
```

Output all the functions included in this module

```
>>> dir(subprocess)
```

Get specific help on a particular function in a module

```
>>> help(subprocess.check_output)
```

This function in the subprocess module will allow you to execute a commandline command from a python program in a list of strings.

This could save you some research time on the web or in books when learning how to use the functions in a module.

2. Import the subprocess module and use the check_output function. Determine if google.com is up and can be reached by your computer. To do this from the commandline you would type the command: `ping -c 5 www.google.com`. Option is `-n` in Windows. Write a small Python program that will take from the user the web url they would like to ping and issue the correct command to determine if that website is reachable by your computer. The output should look something like the following if your computers network is configured correctly and if the website is up and running:
PING www.google.com (74.125.239.17): 56 data bytes
64 bytes from 74.125.239.17: icmp_seq=0 ttl=54 time=24.563 ms
64 bytes from 74.125.239.17: icmp_seq=1 ttl=54 time=24.082 ms
64 bytes from 74.125.239.17: icmp_seq=2 ttl=54 time=31.718 ms
64 bytes from 74.125.239.17: icmp_seq=3 ttl=54 time=24.468 ms
64 bytes from 74.125.239.17: icmp_seq=4 ttl=54 time=24.451 ms
3. Write a program that will calculate the area of a square and the area of a circle. You will ask the user for the length of one leg of the square and the radius of the circle. As a reminder, here are math functions that you will need to implement:

Area of circle = $\pi \times radius^2 = \pi \times radius \times radius$

Area of square = $length \times width$

Lab 6 – str and list

Strings–str

Lists–list

Strings are one of the most popular types in Python.

We can create them simply by enclosing characters in quotes.

- Remember Python treats single quotes the same as double quotes.
- triple quotes allows strings to span multiple lines, including verbatim NEWLINES, TABs, and any other special characters.

Creating strings is as easy as assigning a string to a variable.

For example:

```
var1 = 'Go Pack Go!'
var2 = "Python is not snakes on a plane"
```

Accessing Values in Strings: Donald Keidel, Ph.D. 2016

Python does not support a character type; these are treated as strings of length one, thus also considered a *substring*.

To access *substrings*, use the square brackets for slicing along with the index or indices to obtain your substring.

Following is a simple example:

```
var1 = 'Go Pack Go!'
var2 = "Python is not snakes on a plane"
print "var1[0]: ", var1[0]
print "var2[1:5]: ", var2[1:5]
```

When the above code is executed, it produces the following result:

```
var1[0]: G
var2[1:5]: ytho
```

Remember, to determine all the built-in methods (functions) for str objects do the following:

```
>>> dir(str)
['_add_', '_class_', '_contains_', '_delattr_', '_doc_', '_eq_', '_format_', '_ge_',
'_getattribute_', '_getitem_', '_getnewargs_', '_getslice_', '_gt_', '_hash_',
'_init_', '_le_', '_len_', '_lt_', '_mod_', '_mul_', '_ne_', '_new_', '_reduce_',
'_reduce_ex_', '_repr_', '_rmod_', '_rmul_', '_setattr_', '_sizeof_', '_str_',
'_subclasshook_', '_formatter_field_name_split', '_formatter_parser', 'capitalize',
'center', 'count', 'decode', 'encode', 'endswith', 'expandtabs', 'find', 'format', 'index',
'isalnum', 'isalpha', 'isdigit', 'islower', 'isspace', 'istitle', 'isupper', 'join', 'ljust', 'lower',
'lstrip', 'partition', 'replace', 'rfind', 'rindex', 'rjust', 'rpartition', 'rsplit', 'rstrip', 'split',
'splitlines', 'startswith', 'strip', 'swapcase', 'title', 'translate', 'upper', 'zfill']
```

Looking closer at a few built-in functions:

Remember, when calling a method (function) of any Python object you need to use the dot operator.

```
>>> string = 'Attention future Python developers'
>>>
>>> components = string.split()
>>>
>>> print components    Donald Keidel, Ph.D. 2016
['Attention', 'future', 'Python', 'developers']
>>>
>>> string.upper()
'ATTENTION FUTURE PYTHON DEVELOPERS'
>>>
>>> string.lower()
'attention future python developers'
>>>
>>> ','.join(components)
'Attention,future,Python,developers'
>>>
```

The most basic data structure in Python is the **sequence**.

Each element of a sequence is assigned a number - its position or *index*.
The first index is zero, the second index is one, and so forth.

Python Lists

The list is a most versatile datatype available in Python which can be written as a list of *comma-separated* values (items) between *square brackets*.

Items of a list do not have to be the same type.

Creating a list is as simple as putting different comma-separated values between square brackets.

For example:

```
list1 = ['Chemistry', 'Biochemistry', 1999, 2007 ]
list2 = [12, 87, 27, 52, 90 ]
list3 = ["a", "b", "c", "d"]
```

Like *string* indices, *list* indices start at 0, and lists can be sliced, concatenated and so on.

Donald Keidel, Ph.D. 2016

Listing out all method of list object:

```
>>> dir(list)
['_add_', '_class_', '_contains_', '_delattr_', '_delitem_', '_delslice_', '_doc_',
'_eq_', '_format_', '_ge_', '_getattribute_', '_getitem_', '_getslice_', '_gt_',
'_hash_', '_iadd_', '_imul_', '_init_', '_iter_', '_le_', '_len_', '_lt_', '_mul_',
'_ne_', '_new_', '_reduce_', '_reduce_ex_', '_repr_', '_reversed_', '_rmul_',
'_setattr_', '_setitem_', '_setslice_', '_sizeof_', '_str_', '_subclasshook_',
'append', 'count', 'extend', 'index', 'insert', 'pop', 'remove', 'reverse', 'sort']
```

To get help on how to use the function and other info:

```
>>> help(list.append)
```

Help on method_descriptor:

```
append(...)
    L.append(object) -- append object to end
```

```
>>> list2 = [12, 87, 27, 52, 90 ];  
>>> list2.append(30)  
>>> print list2  
[12, 87, 27, 52, 90, 30]
```

```
>>> list2.reverse()  
>>> list2  
[30, 90, 52, 27, 87, 12]
```

Lists are *mutable*, meaning they can be altered after being created. The *string* object is not.

Donald Keidel, Ph.D. 2016

Lab 6 – Exercises:

1. Ask the user for a list of numbers and then take this list of numbers and sort them.
2. Write a program that will ask the user for a sentence. Have functions in the program that will do the following:
 - a. Create an uppercase version of the sentence and return it
 - b. Create a lowercase version of the sentence and return it
 - c. split the sentence up and place comma between each element and return it
 - d. Take the sentence and turn it into a question by adding a question mark to the end and then return it
3. Write a program that will take anything the user inputs into a list. Have the program stop taking input when the user types 'quit'. Write a function in the program that will determine the length of the list (hint use built-in function `len()`). Write another function that will select a *random* number between 1 and the length of the list. Write another function that will use the built in function for list that will *remove* the randomly chosen element from the list and print out the list.

Donald Keidel, Ph.D. 2016

Homework 3:

Write a program that takes a 3 word phrase and then converts the words to Pig Latin.

To review, Pig Latin takes the first letter of a word, puts it at the end, and appends “ay”. The only exception is if the first letter is a vowel, in which case we keep it as it is and append “hay” to the end.

E.g. “boot” → “ootbay” , and “image” → “imagehay” .

There are many more Pig Latin rules, but for this homework this is sufficient.

Define a GLOBAL list at the top of your code file called VOWELS. This way, you can check if a letter x is a vowel with the expression x in VOWELS. Remember to get a word except for the first letter, you can use word[1:].

It’s tricky for us to deal with punctuation and numbers with what we know so far, so instead, ask the user to enter only 3 words and spaces. You need to convert their input from a string to a list. Also convert the 3 word phrase to all lower when doing comparisons against the vowels list.

Using a list of words, you can go through each word and convert it to Pig Latin by calling the function ConvertWordToPigLatin. Make sure you have a check in your code that it only converts lists that are of length 3 to Pig Latin.

Break all the tasks into functions. Should have functions that: reads user input with name AskUserForSentence, lowercases the sentence called LowercaseSentence, splits the phrase called SplitSentenceIntoList, converts the word to pig latin called ConvertWordToPigLatin and prints out the 3 word phrase in pig latin called PrintThreeWordPhrase. Remember, using functions makes your code much more re-usable, readable and clean.

Once you have your program working, make it interactive such that it keeps translating 3 word phrases into pig latin until the user enters in the phrase QUIT.

References:

1. <http://www.itmaybeahack.com/homepage/books/nonprog/html/index.html>
2. <http://www.geeksaresexy.net/2007/03/20/how-does-the-binary-system-work-an-introduction/>
3. <http://www.tutorialspoint.com/python/>
4. [http://en.wikipedia.org/wiki/IDLE_\(Python\)](http://en.wikipedia.org/wiki/IDLE_(Python))
5. <http://www.python-course.eu/>

Donald Keidel, Ph.D. 2016