Functions

Functions

Remember our PB&J example?

Which is easier?:

1. Get bread
2. Get knife
4. Open PB
3. Put PB on knife
4. Spread PB on bread ...

1. Make PB&J

Functions are a way to *group* instructions.

Functions

What it's like in our minds:

> "Make a peanut butter and jelly sandwich."

In Python, it could be expressed as:

```
make_pbj(bread, pb, jam, knife)
```

function **name**

function **parameters**

Functions

Let's create a function in the interpreter:

```
>>> def say_hello(myname):
...     print 'Hello', myname
```

Remember: The second line should be indented 4 spaces.

Functions

def is the **keyword** we always use to define a function.

'myname' is a **parameter**.

```
>>> def say_hello(myname):
...     print 'Hello', myname
```

Functions

Now we'll *call* the function:

```
>>> say_hello("Katie")
Hello, Katie

>>> say_hello("Barbara")
Hello, Barbara
```

Use your new function to say
hello to some of your classmates!

Functions: Practice

1. Work alone or with a neighbor to create a function that **doubles a number** and prints it out.

2. Work alone or with a neighbor to create a function that takes **two numbers**, multiplies them together, and prints out the result.

Functions: Practice

1. Work alone or with a neighbor to create a function that **doubles a number** and prints it out.

```
>>> def double_number(number):
...     print number * 2

>>> double_number(14)
28
```

Functions: Practice

2. Work alone or with a neighbor to create a function that takes **two numbers**, multiplies them together, and prints out the result.

```
>>> def multiply(num1, num2):
...     print num1 * num2

>>> multiply(4, 5)
20
```

Functions: Output

`print` displays something to the screen. But what if you want to save the value that results from a calculation, like your doubled number?

```
>>> def double_number(number):
...     print number * 2

>>> new_number = double_number(12)
24

>>> new_number
>>>

>>> new_number = double_number(12)
24
```

Functions: Output

```
>>> def double_number(number):
...     return number * 2

>>> new_number = double_number(12)
24

>>> new_number
24
```

Functions

Rules:

- Functions are **defined** using `def`.
- Functions are **called** using **parentheses**.
- Functions take **input** and can return **output**.
- `print` **displays** information, but does not give a value
- `return` gives a **value** to the caller (you!)

Comments

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

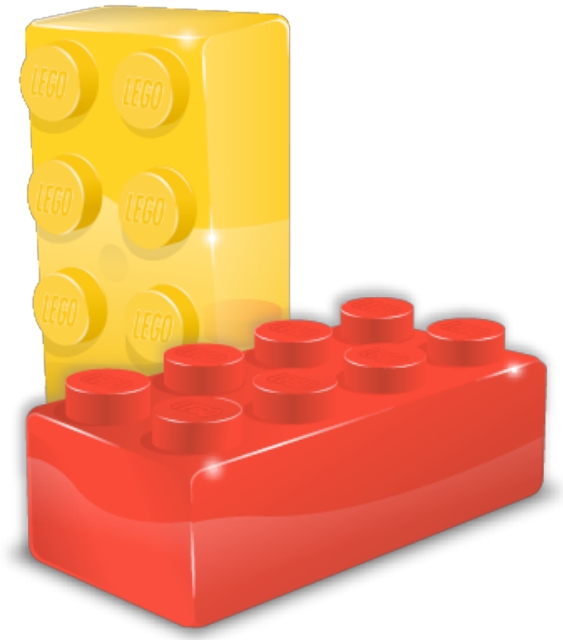
```
>>> def double_number(number):
...     # Here's where we double the number:
...     return number * 2

>>> new_number = double_number(12)
24

>>> # You can also have a comment by itself
```

Modules

Modules



A module is a block of code that can be combined with other blocks to build a program.

You can use different combinations of modules to do different jobs, just like you can combine the same LEGO blocks in many different ways.

Modules

There are lots of modules that are a part of the Python Standard Library

How to use a module:

```
>>> import random
>>> print random.randint(1, 100)

>>> import time
>>> time.time()

>>> import calendar
>>> calendar.prmonth(2013, 3)
```

Modules

A few more examples:

```
>>> import os
>>> for file in os.listdir("~/Desktop"):
...     print file

>>> import urllib
>>> myurl = urllib.urlopen('http://www.python.org')
>>> print myurl.read()
```

You can find out about other modules at: http://docs.python.org