We call these reusable pieces of code “functions”.

There are two kinds of functions in Python.

- Built-in functions that are provided as part of Python - raw_input(), type(), float(), int() ...
- Functions that we define ourselves and then use

We treat the of the built-in function names as "new" reserved words (i.e. we avoid them as variable names)
Function Definition

- In Python a function is some reusable code that takes arguments(s) as input does some computation and then returns a result or results
- We define a function using the `def` reserved word
- We call/invoke the function by using the function name, parenthesis and arguments in an expression

Max Function

```
>>> big = max('Hello world')
>>> print big
'w'
```

```
def max(inp):
    for x in y:
        blah
    blah
```

Guido wrote this code
Type Conversions

- When you put an integer and floating point in an expression the integer is implicitly converted to a float.
- You can control this with the built-in functions int() and float:

```
>>> print float(99) / 100
0.99
>>> i = 42
>>> type(i)
<type 'int'>
>>> f = float(i)
>>> print f
42.0
>>> type(f)
<type 'float'>
>>> print 1 + 2 * float(3) / 4 - 5
-2.5
```
Definitions and Uses

- Once we have defined a function, we can call (or invoke) it as many times as we like
- This is the store and reuse pattern

Arguments

- An argument is a value we pass into the function as its input when we call the function
- We use arguments so we can direct the function to do different kinds of work when we call it at different times
- We put the arguments in parenthesis after the name of the function

```python
big = max('Hello world')
```

Parameters

- A parameter is a variable which we use in the function definition that is a “handle” that allows the code in the function to access the arguments for a particular function invocation.

```python
>>> def greet(lang):
...     if lang == 'es':
...         print 'Hola'
...     elif lang == 'fr':
...         print 'Bonjour'
...     else:
...         print 'Hello'
... >>> greet('en')
Hello
>>> greet('es')
Hola
>>> greet('fr')
Bonjour
```
Return Values

- Often a function will take its arguments, do some computation and return a value to be used as the value of the function call in the calling expression. The return keyword is used for this.

```python
def greet():
    return "Hello"
print greet(), "Glenn"
print greet(), "Sally"
```

Return Value

- A “fruitful” function is one that produces a result (or return value)
- The return statement ends the function execution and “sends back” the result of the function

```python
def greet(lang):
    if lang == 'es':
        return 'Hola'
    elif lang == 'fr':
        return 'Bonjour'
    else:
        return 'Hello'

>>> def greet(lang):
    ...     if lang == 'es':
    ...         return 'Hola'
    ...     elif lang == 'fr':
    ...         return 'Bonjour'
    ...     else:
    ...         return 'Hello'
    ... >>> print greet('en'), 'Glenn'
Hello Glenn
>>> print greet('es'), 'Sally'
Hola Sally
>>> print greet('fr'), 'Michael'
Bonjour Michael
```

Arguments, Parameters, and Results

- We can define more than one parameter in the function definition
- We simply add more arguments when we call the function
- We match the number and order of arguments and parameters

```python
>>> big = max('Hello world')
>>> print big
'w'
def addtwo(a, b):
    added = a + b
    return added

x = addtwo(3, 5)
print x
```
Void (non-fruitful) Functions

- When a function does not return a value, we call it a "void" function
- Functions that return values are "fruitful" functions
- Void functions are "not fruitful"

To function or not to function...

- Organize your code into "paragraphs" - capture a complete thought and "name it"
- Don't repeat yourself - make it work once and then reuse it
- If something gets too long or complex, break up logical chunks and put those chunks in functions
- Make a library of common stuff that you do over and over - perhaps share this with your friends...

Summary

- Functions
- Built-In Functions
- Type conversion (int, float)
- Math functions (sin, sqrt)
- Try / except (again)
- Arguments
- Parameters
- Results (Fruitful functions)
- Void (non-fruitful) functions
- Why use functions?

Exercise

Rewrite your pay computation with time-and-a-half for overtime and create a function called computepay which takes two parameters (hours and rate).

Enter Hours: 45
Enter Rate: 10
Pay: 475.0

475 = 40 * 10 + 5 * 15