functions (part 2) and modules

Ben Bolker

17:51 22 January 2017

Functions calling functions

Functions can call functions (even themselves, **recursively**):

```
def factorial(x):
    if (x==1):
        return(1)
    return(x*factorial(x-1))

factorial(5)
```

What happens if we forget to put in the `if` clause?

- More often, functions call functions
- You can pass anything to a function as an argument (even a function!)

```
def repeat_fun(f,x,n):
    for i in range(n):
        x=f(x)
    return(x)

def sqr(x):
    return(x*x)

repeat_fun(sqr,3,3)
```

Scope

- Where does Python look for things?
- What happens here?

```
z = 1
def add_z(x):
    return(x+z)

add_z(z)
```

- **LEGB** (Local, Enclosing, Global, Built-in)
  - *Local*: symbols defined in the function, and arguments
  - *Enclosing*: symbols defined *in the function within which this function was defined*
  - *Global*: elsewhere in the file/module
  - *Built-in*: Python keywords

Modules

importing

- `import`
- refer to functions via module prefix
- `import VeryLongModuleName as vlmn`: use abbreviation

finding out about

- official modules
- list of useful module
- `math`, `cmath`, `re`, `random`, `numpy`, `scipy`, `matplotlib`, `timeit`