CSC 223 - Advanced Scientific Programming

Python Functions

Functions

- Functions are a way to organize code into reusable pieces.
- Python has two statements for creating functions:
  - `def` for any type of function
  - `lambda` for short anonymous functions

Using Functions

- Functions are called using parentheses.

```
print('abc')
```

- Some functions have *keyword* arguments that are specified by name.

```
print(1, 2, 3, sep="--")
```

- When a function has both positional and keyword arguments, the keyword arguments must be at the end.

Defining Functions

- Functions are defined with the `def` statement.

```
def fibonacci(N):
    L = []
    a, b, = 0, 1
    while len(L) < N:
        a, b = b, a + b
        L.append(a)
    return L
```

- A function with no `return` statement returns `None`.
- Multiple return values can be put in a tuple.

Default Argument Values

- The parameters of a function can be given default values.

```
def fibonacci(N, a=0, b=1):
    L = []
    a, b, = 0, 1
    while len(L) < N:
        a, b = b, a + b
        L.append(a)
    return L
```

- The values of parameters with default values can be changed by calling the function with the extra positional or keyword arguments.

```
>>> fibonacci(10, 0, 2)
[2, 2, 4, 6, 10, 16, 26, 42, 68, 110]
>>> fibonacci(10, b=3, a=1)
[3, 4, 7, 11, 18, 29, 47, 76, 123, 199]
```

Flexible Arguments

- A Python function can take an arbitrary number of positional or keyword arguments.

```
def catch_all(*args, **kwargs):
    print("args =", args)
    print("kwargs =", kwargs)
```

- The * before a variable means "expand this as a sequence"
- The ** before a variable means "expand this as a dictionary"

```
>>> inputs = (1, 2, 3)
>>> keywords = {'pi': 3.14}
>>> catch_all(*inputs, **keywords)
args = (1, 2, 3)
kwargs = {'pi': 3.14}
```

Anonymous Functions

- The `lambda` statement can be used to create a short function

```
add = lambda x, y: x + y
```

- Everything in Python is an object including functions.
- This means that functions can be passed to functions and returned from functions.

```
>>> L = [(1, 4), (2, 3), (3, 2), (4, 1)]
>>> sorted(L, key=lambda x: x[1])
[(4, 1), (3, 2), (2, 3), (1, 4)]
```

Function Documentation

- Python uses a *docstring* for function documentation
- Docstrings must be indented with the function body

```
def pressure(v, t, n):
    """Compute the pressure in pascals of
    an ideal gas

    v -- volume of gas, in cubic meters
    t -- absolute temperature in degrees kelvin
    n -- particles of gas
    """
    k = 1.38e-23 # Boltzmann's constant
    return n * k * t / v
```