CISS433: Python Programming
Lecture 11: Function

Yihsiang Liow

Agenda

- Study functions

Function

- Format:

```
def <func>(<parameters>):
    <stmt>
```

- Here's a boring standard textbook example:

```
def savings(principal, interest, num_years):
    yr = 1
    while yr <= num_years:
        principal *= (1 + interest)
        yr += 1
    return principal

print(savings(1000, 0.01, 10))
```

Rewrite it using `for` instead of `while`.

Recursion

- Python supports recursion
- To protect against infinite recursion, there is a maximum recursion depth. This can be tweaked if necessary.
- Here's the mandatory fibonacci:

```
def fib(n):
    if n==0 or n==1:
        return 1
    return fib(n - 1) + fib(n - 2)
print("Here's fibo of 10:", fib(10))
```

Recursion

- Exercise. Write a recursive function `factorial` that computes the factorial of a positive integer. If the integer passed in is negative, return 1.
- Exercise. Write a word count function that accepts a string and returns the number of words in the string.

Returning Multiple Values

- Example:

```
def f():
    return 1, 2, 3


x, y, z = f()
print(x, y, z)
```

- Actually the thing that is returned is the tuple `(1, 2, 3)`. In other words `1, 2, 3` is a short hand notation for `(1, 2, 3)`.
- Python can "unpack" tuples:

```
x, y, z = 42, 43, 44
```

Default Values

- **Example.**

```
def hello(name="Arnold Schwarzenegger"):
    print("Hello", name)

hello()                # use default value
hello("John Nash") # overwriting default
```

Keywords & Default Values

```
def happy_bday(name="", age=0):
    if name=="" and age==0:
        pass # no-op, better: invert logic
    else:
        print("Happy b'day", name)
        print("You're", age," years old")

happy_bday()
happy_bday("Susan", 10)
happy_bday(age=-1)
happy_bday(age=200, name="John") # magic
```

Nestation

- Function definitions can be nested. (Recall: You cannot nest function definitions in C/C++.)
- Try:

```
def f():
    def g(): print("hi")
    g()
f()
```

Warning

- Try this out. Explain what's happening.

```
def f(a):
    a.append(1)

def g(a):
    a = 0

x = [1,2,3]
f(x)
print(x)
y = 1
g(y)
print(y)
```

Miscellaneous

- Functions are objects:

```
def f():
    print("Hello, World!")
g = f
print(type(g), type(f))
print(id(g), id(f))
```

- Try this too:

```
x = 0
if x==0:
    def f(): print 0
else:
    def f(): print 1
f()
```

Miscellaneous

- The usual rules apply to functions. For instance variables created in a function is local to the function.
- We will cover scope rules in detail later when we cover modules.