FINE 434: FinTech

Lecture 5

Professor Fahad Saleh

McGill University - Desautels

Introduction

You might have considered the situation where you would like to reuse a piece of code, just with a few different values. Instead of rewriting the whole code, it's much cleaner to define a function, which can then be used repeatedly.

```
def function_name(x,y,...):
    # Code Here
```

Introduction

You might have considered the situation where you would like to reuse a piece of code, just with a few different values. Instead of rewriting the whole code, it's much cleaner to define a function, which can then be used repeatedly.

```
def function_name(x,y,...):
    # Code Here
```

Always start with the keyword def then the function name then the arguments (within parenthesis) then a colon. Then, indent and start coding the function.

Calling a Function

```
def hello_world():
    print("Hello World")
def power(b,x):
    print(str(b)+"^"+str(x)+" = "+str(b**x))

hello_world() # calls hello_world

power(3,2) # calls pow
```

What does this code produce?

```
def hello_world():
    print("Hello World")

def power(b,x):
    print(str(b)+"^"+str(x)+" = "+str(b**x))

helloworld() # calls hello_world

power() # calls pow
```

What does this code produce?

```
def hello_world():
    print("Hello World")

def power(b,x):
    print(str(b)+"^"+str(x)+" = "+str(b**x))

helloworld() # calls hello_world

power() # calls pow
```

What does this code produce?

What's wrong?

Example

```
def knock_knock():
    print(”Knock Knock”)
    print(”Who’s there?”)
print(”iono...”)
```

What does the call knock_knock() produce?

return

A function may **return** something. To have a function return, just use the keyword return at the end of the function.

e.g.

```
def power(b,x):
    return b**x

result = power(3,2)
print("3 ^ 2 = "+str(result))
```

Example

```
def add(a,b):
    x = a
    return x
    x = x + b
    return x
y = add(1,2)
```

What is the value of y?

Explicit References

Given a function f(a,b), calling f(b = x, a = y) gives f(y, x).

```
1  def subtract(a,b):
2      return a - b
3
4  print(subtract(4,2))
5
6  print(subtract(b=4,a=2))
```

```
2
-2
```

Defaults

When defining a function, one can specify a default input. In such a case, the user may omit the input entirely. The default is specified in the function declaration.

e.g.

```
def subtract(a,b=1):
    return a - b
```

The default value for b is 1.

Your Turn

```
def subtract(a,b=1):
    return a - b
```

What do the following calls produce?

```
subtract(4)
```

```
subtract(a=4)
```

```
subtract(b=4)
```

Application: Integration

Write a function to find I such that $\left| \text{I} - \int_0^1 2^{x^2} dx \right| < \text{tol}$ with $\text{tol} > 0$ being an argument.

i.e.

```
def integral(tol):
    # Your code here
```

Application: Integration

```
def integral(tol): # INEFFICIENT!
    partitions = 1
    integrand = 1
    max_error = 2
    while max_error > tol:
        partitions = partitions*1000
        min_integrand = 0.0
        max_integrand = 0.0
        for p in range(0,partitions):
            x_l = float(p)/float(partitions)
            x_u = x_l + 1.0/float(partitions)
            min_integrand = min_integrand + 2**(x_l**2)
            max_integrand = max_integrand + 2**(x_u**2)
        max_integrand = max_integrand/float(partitions)
        min_integrand = min_integrand/float(partitions)
        max_error = max_integrand - min_integrand
        integrand = (max_integrand + min_integrand)/2.0
    return integrand
integral(.0001)
```

```
1.288226364306182
```