MATH 365
Functions

1 Introduction

In many cases, you will want to evaluate expressions multiple times using different values of the variables. While you might be able to cut and paste the expressions multiple times, this is error prone. It is much better to create a "function" that can be called multiple times using different arguments.

For example from the Loops and Conditional Statements lab we could simplify the standard deviation calculation

```
mu = 0
for i = 1:n
    mu = mu + x(i)
end
mu = mu/n
s=0
for i = 1:n
    s = s + (x(i)-mu)^2
end
s = sqrt(s/(n-1))
```

Instead define

```
function mymean(x)
    n=len(x)
    mu = 0
    for i = 1:n
        mu = mu + x(i)
    end
    mu = mu/n
return mu
end
```

so that the standard deviation calculation becomes

```
mu=mymean(x)
s=0
for i = 1:n
    s = s + (x(i)-mu)^2
end
s = sqrt(s/(n-1))
```

Once a function is defined it can be used anywhere, including in other functions. For example, we could also define a function for the standard deviation that uses the function we created for the mean:

```
function mystd(x)
    mu=mymean(x)
    n=len(x)
    s=0
    for i = 1:n
        s = s + (x(i)-mu)^2
    end
    s = sqrt(s/(n-1))
return s
end
```

so that we only need to type

```
s=mystd(x)
```

to get the standard deviation.

You will notice that I did not call these functions *mean* and *std*. That is because many libraries already have functions with those names and it can cause errors to use them in a different way.

Some programming languages make a distinction between "functions" that return values and "subroutines" that do not return anything but rather *do* something like produce a plot. In Python and MATLAB there is only one kind, functions, and they can return single, multiple, or no values at all.

Argument variables within functions exist in their own namespace. This means that assignment of an argument to a new value does not affect the original value outside of the function. For example using the function mymean above if we type

```
mu=5
x=[1, 1, 1, 1]
xbar=mymean(x)
print(mu)
```

the answer will be 5, which has nothing to do with the mean of x. On the other hand, print(xbar) will produce the mean of x which is equal to 1.

1.1 Anonymous Functions

Anonymous functions are in-line functions that can be generated on the fly to accomplish some small task. You can assign them a name, but you don't need to; hence, they are often called anonymous functions. You may find them convenient to send a function like $f(x) = \cos(x)$ into a named function. Most likely the named function will be the important part of the code while $f(x)$ just tests it. If you use an anonymous function for $f(x)$ then you wont clutter your code with a lot of one line statements.

1.2 Functions in Python

The function to calculate the average that we wrote in pseudocode in the previous section is written in Python as

```
def mymean(x):
    n=len(x)
    mu = 0
    for i in range(n):
        mu = mu + x[i]
    mu = mu/n
    return mu
```

The **return** statement causes a function, loop or conditional to exit or terminate immediately, even if it is not the last statement of the function, loop or conditional. The return statement also identifies what variables should be passed out. If no return statement is present within a function, or if the return statement is used without a return value, Python automatically returns the special value None when the function is called.

To return both the mean and the standard deviation, we would also create the function

```
def mystd(x):
    mu=mymean(x)
    n=len(x)
    s=0
    for i in range(n):
        s = s + (x[i]-mu)**2
    s = (s/(n-1))**(1/2)
    return mu, s
```

and call it with

```
xbar, sigma = mystd(x)
```

Note that if we typed the statement

```
print(s)
```

we would get the error statement *NameError: name 's' is not defined.*

An **anonymous function** is also called a lambda expression so Python uses the general form

```
lambda arg1, arg2, ... : output
```

The arguments arg1, arg2, ... are inputs to a lambda, just as for a functions, and the output is an expression using the arguments. For example, these two functions are equivalent in Python

```
def f(a, b):
    return 3*a+b**2
```

```
g = lambda a, b : 3*a+b**2
```

i.e. f(a,b)=g(a,b).

2 Individual Lab

Please answer the following questions in Blackboard Part 1:

1. What value does this function return if you pass in 4?

```
function fact(N)
    var =1
    while N > 0
        var = var*N
        N = N-1
    end
    return var
end
```

2. What value does this function return if you run mystery(2,5)?

```
def mystery(X, Y):
    if X = Y :
        return X*Y
    if X > Y:
        return X - Y
    else:
        return Y - X
```

3. What statement will correctly find the slope of the line passing through the points (1,-2) and (-1,2)?

```
def myslope(x1,x2,y1,y2):
    slope=(y2-y1)/(x2-x1)
    return slope
```

4. Define the functions

```
def fun_f(x):
    fun=np.cos(x)
    return fun
```

```
def fun_g(x):
    fun=x**2
    return fun
```

and identify how to evaluate $\cos(x^2)$, $(\cos(x))^2$, $x^2\cos(x)$ and $x^2 + \cos(x)$.

5. Review Netwon's method from Calculus I: `http://tutorial.math.lamar.edu/Classes/CalcI/NewtonsMethod.aspx` and answer the question in Blackboard.

6. Review Taylor series from Calculus II: `http://tutorial.math.lamar.edu/Classes/CalcII/TaylorSeries.aspx` and answer the question in Blackboard.

7. Review numerical integration from Calculus II: `http://tutorial.math.lamar.edu/Classes/CalcII/ApproximatingDefIntegrals.aspx` and answer the question in Blackboard.

Part 2:

1. What is the nth degree Taylor series polynomial for e^x?
2. Assume we want to find the roots of $f(x) = x^3 - 7x^2 + 8x - 3$. Start with an initial guess of $x_0 = 5$ and find the first estimate x_1 using Newton's method.
3. Approximate the derivative of $f(x) = \sin(x)$ at $x = 1.0$ with the formula $\frac{f(x+h)-f(x)}{h}$ and a step size $h = 0.1$.
4. Approximate $\int_1^5 \frac{1}{x^3+1}\, dx$ with midpoint rule. Divide the interval into 4 subintervals of equal length in your approximation.

3 Group work

1. Identify which function does (i) Netwon's method, (ii) Taylor series, (iii) numerical differentiation, or (iv) numerical integration. Please write your answer on the supplied note cards e.g. func1 and Newton, func2 and Taylor, etc.

```
def func1(f, x, a,b,h):
    x=np.arange(a,b,h)
    g = (f(x+h) - f(x))/h
    return g
```

```
def func2(n,x):
    f = 0
    for i in range(n):
        f+= x**i/np.math.factorial(i)
    return f
```

```
def func3(x, f, fprime, max_iter, tol):
    for i in range(max_iter):
        step = f(x)/fprime(x)
        if abs(step) < tol:
            return i, x
        x -= step
    return i, x
```

```
def func4(f,a,b,h):
    sum = 0.0
    x = a + h/2
        while (x < b):
            sum += h * f(x)
            x += h
        return sum
```

2. Please form a new group of 3-4 people. Each team will consist of:
   - The **manager**, responsible for coordinating the work of the team. The manager is the person who has the most siblings.
   - The **spokesperson**, who will report your group's work to the rest of the class. The spokesperson is the person who has the shortest time to graduation.
   - The **scribe**, who will be responsible for writing down your team's findings. The scribe is the person who has taken the most foreign language courses.
   - The **timekeeper**, who will keep track of time for each exercise. The timekeeper is the person who was not assigned a role already. In case of a three-person group, the manager takes the role of a timekeeper.

One person should not have more than one role (except for the manager/timekeeper combo), so if a person who already was assigned a task meets the criteria for another one, the person who is next in line (i.e., second-longest commute) takes this role.

Each group will be assigned one method and the spokesperson will present their answers to the following questions to the class.

(a) Newton's method

i. Explain the inputs and outputs to the supplied function and their dimension e.g. a scalar or a vector with specific dimension.

ii. Describe x and fprime in the supplied function and their role in Newton's method.

iii. Describe the role of max_iter and tol in the supplied function. Explain what happens when they are adjusted.

iv. In the supplied function, there is an initial guess of the root, but it is implicitly defined. Explain.

v. Give a specific example where you call the function and print or plot the output.

(b) Taylor series

i. Explain the inputs and outputs to the supplied function and their dimension e.g. a scalar or a vector with specific dimension.

ii. Describe the role of n. Give an example to describe the effect of it being adjusted.

iii. If the supplied function can be used to find Taylor series of any mathematical function, give an example. If it can't be used for any mathematical function, how would you adjust the function to estimate the Taylor series for a different mathematical function?

iv. Give a specific example where you call the function and print or plot the output.

(c) Numerical differentiation

i. Explain the inputs and outputs to the supplied function and their dimension e.g. a scalar or a vector with specific dimension.

ii. What do a,b, and h represent? What is the effect of adjusting h?

iii. Give an example of how you would input a specific f.

iv. How could you change the differentiation function so that it takes data rather than a function as input? What would be potential problems/errors?

v. Give a specific example where you call the function and print or plot the output.

(d) Numerical integration

i. Explain the inputs and outputs to the supplied function and their dimension e.g. a scalar or a vector with specific dimension.

ii. Describe what happens in the supplied function when we input h<0.

iii. Explain why this is called the midpoint rule.

iv. Give a specific example where you call the function and print or plot the output.