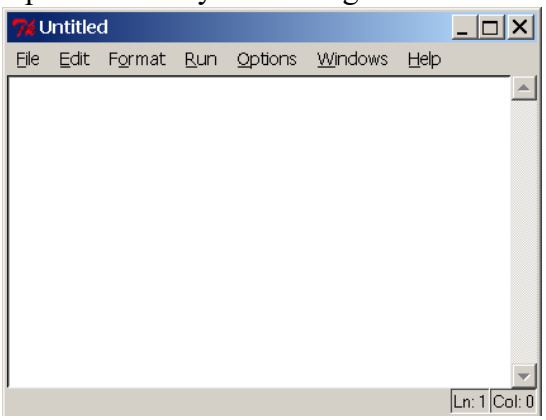
5.1 Computational Toolbox—Tools of the Trade: *Python* Tutorial 1

Starting Python:

Python comes with a tool called IDLE. IDLE can be thought of as a **shell** running an **interpreter**, as an **editor** or as an integrated development environment (IDE.).

Open IDLE. If you do not get a window that looks like this:



then go to FILE → NEW WINDOW

We will type commands into this window, save it, and when we choose RUN → RUN MODULE we will watch as python executes all the commands.

Python comes with lots libraries for special purposes. Because we will be doing scientific computing, it is convenient to have the library of mathematical functions available to us. Look at the following example:

```
import math

print math.sin(4)
```

The first statement gives us access to the **math** library and the second prints the value of sin(4). Notice that we precede the name of the mathematical function *sin(x)* with the name of the library. Don't forget the "dot".

You can find a listing of all the constants and functions that are available at http://docs.python.org/lib/module-math.html

Numbers and Arithmetic

Try adding the various items that are introduced below to your file, then save and run it to make sure you understand how it works. The `print` command prints the result of a calculation to the screen.

```
print 5/8
print 5/8.0  #or print 5/float(8)
print 5*8

print (1/2.0)**2

print 5 * math.sin(math.pi/2)
```

After you have tried this, here are some things to note:

- The operators are +, -, / , * , and ** for exponentiation and you must use them. 5(2+3) is not a legal statement.
- If you use two integers, python assumes you want an integer as an answer and throws away the decimal part.
- # introduces a comment and python ignores the rest of the line
- The function **`float`** converts its argument to a decimal number

Quick Review Question: Figure out how to calculate e^x which is Euler's number e to the x power. Add this to your file.

Variables and Assignments

We can use variables to store values for future use. Variable names must begin with a letter of the alphabet but then can have any number of letters digits and underscores (_). Python is case sensitive so F1 and f1 are different names. The typical format of an assignment statement is ***var = expression***

```
ans = 5 * math.pi

ans = ans + 1

print ans + 2
```

The *semantics* of the assignment statement is that the expression on the right hand side of the equals sign is evaluated and the value is stored in the variable named on the left hand side. Thus the second line does not say that **`ans`** is equal to one more than itself but that one should be added to the current value of **`ans`** and then stored in **`ans.`** Note that the last line does not change the value of **`ans.`** If a variable appears on the right hand side but has not been given a value yet, python will print an error message.

Quick Review Question: Assign the value 5 to a variable *vel* and then write the correct code to print the value of $\left(7+\frac{vel}{3}\right)^2$.

User Defined Functions

We will often want to define and use functions besides those which exist in libraries. Here is how one would define and use a function which corresponds to the mathematical function $f(x) = x^2 + 1$

```
def f(x):
    ans = x**2 + 1
    return ans

print f(4)
print f(2.5)
```

Note the syntax which requires the reserved word **`def`** and the colon. You must also indent the code because this is how python understands what lines belong to the function. The **`return`** statement determines what the value of the function will be for a given input represented by **`x`** in this definition. Below the definition, you see several calls to the function. **`f(4)`** is evaluated by substituting **`4`** for the **`x`** in the definition. The return value will be 17 and that is what is given to the print function.

Functions can have more than one variable.

```
def sumSquares(x,y):
    ans = x**2 + y**2
    return ans

print sumSquares(3,4)
```

Quick Review Questions:

a) Define and test a function that would allow you to compute $p(t) = 100e^{0.1t}$. Evaluate for t = 12

b) Define a function $pop(initPop,\ r,\ t) = initPop\, e^{rt}$. Then call your function with initPop = 1000, r = 0.15 and t = 12

Printing

You have seen print statements in examples above, but they can be done more generally to label your output or to print multiple values. Anything that appears between the quotes is called a string.

```
x = 3.0
print "x =  ", x, "  and its square is  ", x**2
```

Loops and Lists

It is often necessary to execute a section of code a number of times. Such a block of code is called a ***loop***. In python this is often done with the help of a **list.**

A python list is enclosed in brackets that contain 0 or more items separated by commas. For example

```
list1 = [1,3,5.0]
list2 = [1, "one", 2, "two"]
list3 = []
print list1, list2, list3
```

Notice that a list can have members of different types: integers, strings, and floats can all appear is the same list. It can even be empty. There is a built in function called **range** which returns certain kinds of lists. Try each of the following statements

```
list4 = range(15)
list5 = range(5,15)
list6 = range(4,15,3)

print list4, list5, list6
```

The first form produces a list of integers starting at 0 and including all the subsequent integers up to but not including the parameter. Thus the first example has all the integers from 0 to 14 inclusive. The second form is similar except that it starts at the first parameter instead of at 0. Finally, the third form uses the third parameter as a step size. So the list starts at the first parameter and continues adding the step size until it reaches or exceeds the second parameter.

Quick Review Exercise:
Find a command which will use the range function to produce the list [3,7,11,15]

This is the common way to execute a loop when we know how many times we want to do it. For example, suppose we want to start a variable *dist* at 0.0 and then increment it by 0.25 and print the new value 7 times

```
dist = 0.0
for item in range(7):
    dist= dist+.25
    print dist
```

This is called a ***for loop*** in python and has the following semantics:

`range(7)` is of course the list `[0,1,2,3,4,5,6]` Notice it has 7 members. `item` is the name of a variable (you can name it whatever you like) which will start at the first member of the list (0) go through the body of the loop doing whatever is prescribed. When it reaches the bottom, python checks if there are any more items in the list and if so, item is set equal to the next one and proceeds through the body of the loop again. Eventually, there will be no more elements in the list and the loop will be complete. Notice that the value of item was never used in the loop – its purpose was just to count the number of times the loop has been executed.

Here is another example where the variable is used in the body of the loop.

```
for i in range(10):
    print i, "  ", i**2
```

Quick Review Question: Write the lines of code that will initialize a variable to 1 and then double and print the variable twenty times.

Note that a loop can contain any number of statements. But they all must be indented properly so python knows which statements belong to the loop.