Lab 1: NUMERICAL COMPUTATION IN PYTHON

Before we can begin applying Python to Calculus, we need an introduction to the symbolic package of Python, sympy, as well as some basics of computation, output, and variables.

EXAMPLE:

An open-top box is made by cutting out equal square corners of an 9 x 12 inch sheet of cardboard and folding up the flaps (see similar picture in 1.1 #63). Find the volume of the box formed when the squares have a side length of

a) 1 inch

b) 0.5 inches

c) 3.1415926 inches

Method 1: Python as a calculator

Cutting one inch from all four corners of the cardboard creates a base of 7 x 10 inches with flaps 1 inch high.

Cutting 0.5 inches from all four corners of the cardboard creates a base of 8 x 11 inches with flaps 0.5 inches high.

And, finally, cutting 3.1415926 inches from all four corners of the cardboard creates a base of... 9-(2)(3.1415926) x 12-(2)(3.1415926) inches with flaps 3.1415926 inches high.

In [2]:
```
# Answers to all questions.  Recall the "#" for comments ignored by Python
7*10*1
8*11*0.5
(9-2*3.1415926)*(12-2*3.1415926)*3.1415926 #Pay attention to order of opera
tions here!
```

Out[2]:
```
48.793730470538755
```

What happened here? We did three computations but only got one output? Python, in a sense, only keeps the last computation it was given (this should be obvious here) unless you force it to do otherwise. The **print** command is how to force output

In [3]:
```
# Answers to all questions.  This time printing each answer.
print(7*10*1)
print(8*11*0.5)
print((9-2*3.1415926)*(12-2*3.1415926)*3.1415926)
```

```
70
44.0
48.793730470538755
```

Now we have 3 answers, but no explanation. And, as future engineers, you will learn that explanation is essential to presenting solutions to problems! Fortunately, you can put explanatory text in your print statement using quotes. Notice Python does NOT care whether you use single quotes or double quotes!

Also, we can simplify our typing on the last part by defining x=3.1415926 and just typing "x" in our computation.

In [4]:
```
#Answers to all questions, now with explanations in our output.
print('A flap 1 inch square produces a volume of',7*10*1,'cubic inches.')
print("A flap 0.5 inches square produces a volume of",8*11*0.5,"cubic inches.")
#Now define x=3.1415926
x=3.1415926
print('A flap',x,'inches square produces a volume of',(9-2*x)*(12-2*x)*x,'cubic inches.')
```

```
A flap 1 inch square produces a volume of 70 cubic inches.
A flap 0.5 inches square produces a volume of 44.0 cubic inches.
A flap 3.1415926 inches square produces a volume of 48.793730470538755 cubic inches.
```

Method 2: Using variable expressions and substituting

This problem doesn't have long, drawn out computations, but if it did, it would be better not to have to retype the computation every time you did a new one! Based on the patterns in Method 1, you can generalize by saying: Cutting *x* inches from each corner will produce a base of (9-2x) by (12-2x) with flaps x inches high. Therefore, we will define an expression (call it *Volume*) which is the product of these three dimensions.

In [5]:
```
Volume=(9-2*x)*(12-2*x)*x
```

No output was produced, so let's print Volume and see what it says

In [6]:
```
print(Volume)
```

```
48.793730470538755
```

This was our last answer, which makes sense because we defined x to be 3.1415926. However, we'd rather have an expression for *GENERAL* dimension x. So let's clear the variable and see what happens. (NOTE: I just did a Google search and found *del* clears, i.e. deletes a variable).

In [9]:
```
del x
Volume=(9-2*x)*(12-2*x)*x
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-9-70d04e73c5cc> in <module>
      1 del x
----> 2 Volume=(9-2*x)*(12-2*x)*x

NameError: name 'x' is not defined
```

Now an error. If we read the error message, we sometimes (not always) get a clue as to WHY we got an error. Python treats variables as "storage spaces" for numbers (or many other types of objects, as mentioned in the Course Overview). But we want Python to treat x as, well, the variable x. Enter the symbolic package of Python.

In [10]:
```
from sympy import *  # This "brings in" all the commands from the sympy lib
rary
x=symbols('x')
Volume=(9-2*x)*(12-2*x)*x
print(Volume)
```

```
x*(9 - 2*x)*(12 - 2*x)
```

NOW we have a symbolic expression with a variable in it. We can now substitute values in for x and produce our desired answers. Notice the syntax for this: *Variable.command(arguments)*. This is VERY standard Python syntax!

In [11]:
```
print('A flap 1 inch square produces a volume of',Volume.subs(x,1),'cubic i
nches.')
print('A flap 0.5 inches square produces a volume of',Volume.subs(x,0.5),'c
ubic inches.')
#As before, let's define a variable c=3.1415926 to simplify typing.
c=3.1415926
print('A flap',c,'inches square produces a volume of',Volume.subs(x,c),'cub
ic inches.')
```

```
A flap 1 inch square produces a volume of 70 cubic inches.
A flap 0.5 inches square produces a volume of 44.0000000000000 cubic inche
s.
A flap 3.1415926 inches square produces a volume of 48.7937304705388 cubic
inches.
```

Method 3: Substitute ALL at once!

This is a preview of things to come. Python allows you to make MULTIPLE substitutions at once by creating a list of values (note the [] to create a list) and using a **for** statement (also known as *List Comprehension*). First, define the list of inputs.

In [12]:
```
xvals=[1,0.5,3.1415926]
Volumes=[Volume.subs(x,i) for i in xvals]  #Think of this as "go through th
e list and substitute each value for x"
print('Squares with side lengths',xvals,'produce volumes of',Volumes)
```

```
Squares with side lengths [1, 0.5, 3.1415926] produce volumes of [70, 44.00
00000000000, 48.7937304705388]
```

Of course, we can organize the output a little better, but we'll save that for another time.

One last reminder from the Course Overview: all of the inputs were floating point decimals, so our answers were floating point decimals as well. Suspending practicality for a moment, suppose we wanted to cut a square of side length square root of 2...

In [13]:
```
c=sqrt(2)
print('A flap',c,'inches square produces a volume of',Volume.subs(x,c),'cub
ic inches.')
```

```
A flap sqrt(2) inches square produces a volume of sqrt(2)*(9 - 2*sqrt(2))*
(12 - 2*sqrt(2)) cubic inches.
```

Recall from the Course Overview: Python will produce EXACT answers unless you give floating point decimals as input. We have two options here:

1) We can **simplify** the expression

2) We can convert the expression to a floating point decimal approximation using **evalf**

Let's demonstrate by assigning the value to a new variable, Vanswer

In [14]:
```
Vanswer=Volume.subs(x,c)
print('The exact volume is',Vanswer.simplify())
print('The approximate volume is',Vanswer.evalf())
```

```
The exact volume is -84 + 116*sqrt(2)
The approximate volume is 80.0487732352790
```

In [ ]:
```

```