Alex Hoffer
10/12/15

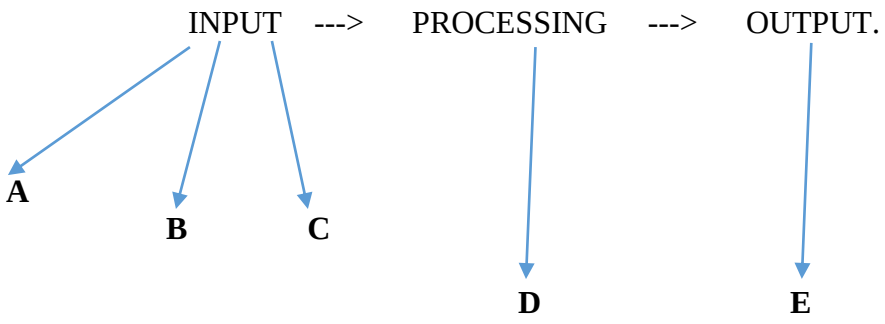
Design Document for Python Calculator

Step 1:

I am to create a calculator using the Python operators. This assumes I know what they all do. The example code on page 1 shows a forced int conversion of the numbers it accepts as variables, however, the example code on page 2 shows a forced float conversion. I will be pulling executive on this one and making mine a forced float conversion, because it doesn't make much sense to include a division operator and not allow the user to find out the true value of x / y when it ends in a decimal. Also unlike the sample code on page 2, the directions do not require that I use a loop to allow the user to enter as many tuple combinations of numbers as they wish. The program is simple, I figure I will merely use if/else statements to figure out what the operand the user wants is, and then use that operator to do answer = (first number entered) *operand* (second number entered).

Step 2:

There will be three primary steps:



INPUT:

a) Establish variable called operator, prompt user for one of the operators, transcribe this input into variable.

b) Establish variable called num1, prompt user for this first number (left operand), store it in the variable.

c) Do the same thing but for num2, which will be the second number (right operand).

PROCESSING:

d) Figure out which operator was entered by the user by creating a column of If-Else statements. The structure should be: "If/Elif (user operator) == Python-Defined Operator(i)" where i corresponds to +, -, %, etc. Recall If applies if it is the first truth statement to be evaluated, otherwise the statement will begin with Elif. *Only one of the statements can evaluate to true*. That statement which evaluates to true will cause a variable to be created. That variable will have (num1) *successfully matched operator* (num2) stored in it.

OUTPUT:

e) Print the variable which holds the answer to the calculation in it.

Step 4:

Note: num1 = x, num2 = y

Value	**Expected**	**Actual meet expected**
y = 0 in x/y	ERROR: Divide by 0.	Yes.
y = -n in x ** y	Solution is a decimal.	Yes (.5 for 2 ** -1)
x = m, y = -n where abs(n) > abs(m) in x + y	Solution is negative.	Yes (-5 for 5 + (-10))
x is a non-integer, y is a non-integer, x % y	Like C/C++, no result, because the mod operator only works with integers.	No. The mod operator in Python apparently works for floats: 4.5 % 2.3 = 2.2.
Entering operator not offered, //.	The program will accept the operator, and then will transcribe the two numbers. However, once it reaches the	Yes

	If/Else mountain it will throw an error.	
Entering data type not supported in float conversion, i.e. a string, for a number.	The program will immediately throw an error because it is tasked with converting the supposed number to a float instantaneously- it receives a string and becomes confused.	Yes.
An erroneous space before typing the initial operator, i.e. instead of "%" but " %".	The program will be sensitive to the space- will search among the If/Else mountain in vain, then throw an error.	Yes. The user must type in the value *exactly* how the program wants it.