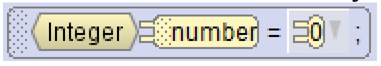
Calculations

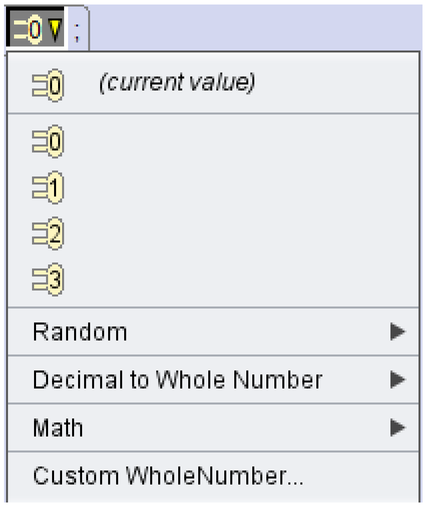
You probably realize that in business applications there is a lot of math. You might be surprised to learn that there is also a lot of math in games and animation. In Alice you might need to know how high the rabbit needs to jump to land on top of a stack of boxes. You might want to know how far a fish needs to swim to hide in the cave and how fast he needs to swim! Even something as simple as adding up points for a game uses math.

Building a math Expression

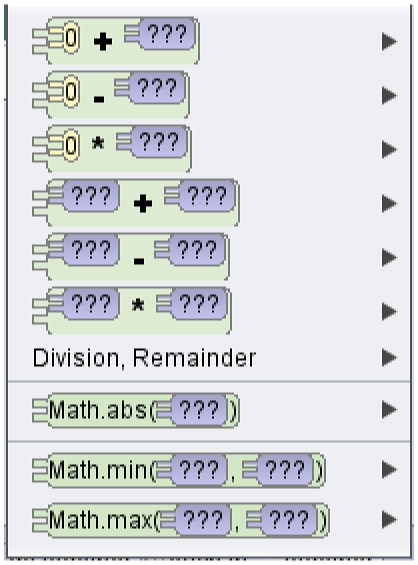
Let's suppose that you have created a variable named num. You selected a value of 0 initially.

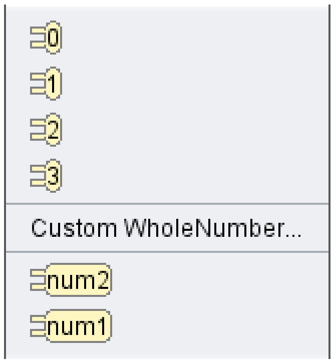


Now you would like to give it a different value: If you click on the 0 you will have these choices:



Now if you click on Math you get the following choices:





The first three use 0 because that is the initial value that you gave to number.
0+??? lets you add a value: if you follow the arrow on the right you can select 0,1,2,3, a custom whole number or an integer variable.

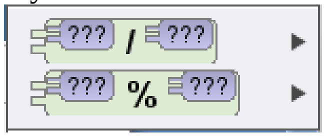
You can continue to make selections and build rather complex math expressions. Usually it is better to use extra variables and do the math one step at a time.

The Math operators

Alice has the following operators:

+	add
-	minus
*	multiplication

If you click on Division/Remainder you will see these choices:



```
??? / ???
??? % ???
```

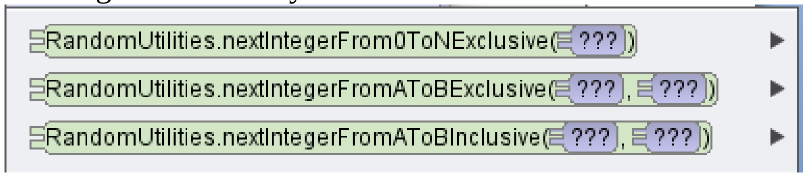
The slash / is division. Be aware that if you divide an integer by an integer you will get an integer result: 3/5 is 0 and 7/5 is 1. There is no decimal and there is no rounding.

The % operator is used to find the remainder. Before children learn about decimal numbers, they may give the answer to division problems as: "17 divided by 5 is 3 with a remainder of 2" Note that 17/5 results in 3, while 17 % 5 results in 2.

Functions

You may have noticed that there are several other choices. The other choices are functions. A function returns a value. The returned value will be used in any calculation. Most functions require arguments. Arguments are the values that the function operates on.

Selecting Random lets you select 1 of 3 functions for random numbers:



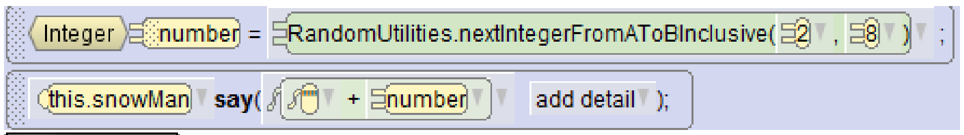
```
RandomUtilities.nextIntegerFrom0ToNExclusive(???)
RandomUtilities.nextIntegerFromAToBExclusive(???, ???)
RandomUtilities.nextIntegerFromAToBInclusive(???, ???)
```

The function RandomUlitites.nextIntegerFrom0ToNExclusive requires one argument: the value for N (the upper limit). If we use the value 5 here it will return a random number from 0 to 4.

The function RandomUlitites.nextIntegerFromAToBExclusive requires two arguments: the value for A (the lower limit) and the value for B (the upper limit). If we use the values 2 and 8 here it will return a random number from 3 to 7.

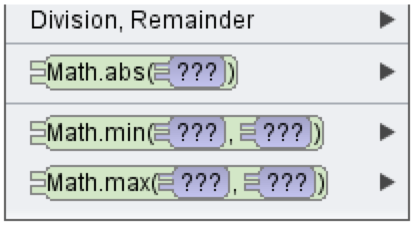
The function RandomUlitites.nextIntegerFromAToBInclusive requires two arguments: the value for A (the lower bound) and the value for B (the upper bound). If we use the values 2 and 8 here it will return a random number from 2 to 8.

You can test any of these functions by assigning a value and then have an actor "say" it.



```
Integer number = RandomUtilities.nextIntegerFromAToBInclusive(2, 8) ;
this.snowMan say( "" + number  add detail );
```

There are also 3 functions shown below Division / Remainder:

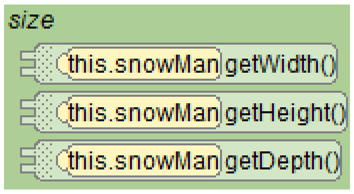


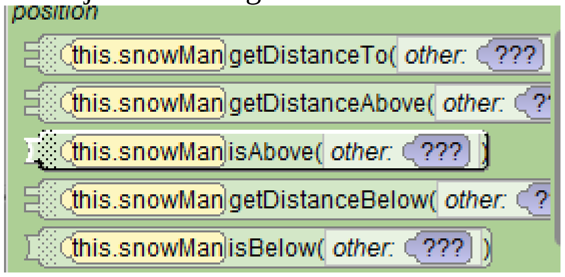
Math.abs(???) requires one argument it returns the absolute value (a positive integer). For example Math.abs(5) returns 5. Math.abs(-5) also returns 5.

Math.min(???, ???) requires two arguments and returns the minimum (smaller) of the two arguments. For example Math.min(5, 3) returns 3.

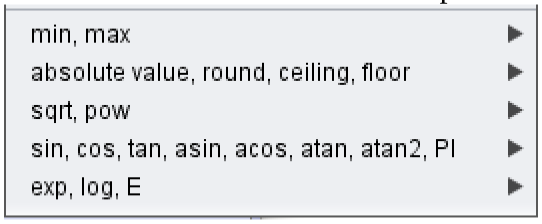
Math.max(???, ???) requires two arguments and returns the maximum (larger) of the two arguments. For example Math.max(5, 3) returns 5.

You will find other functions on the functions tab. A few examples are shown below.
The function getWidth does not have any arguments. It returns the width of the object (a double).
The function isAbove requires an object as the argument and return a boolean.



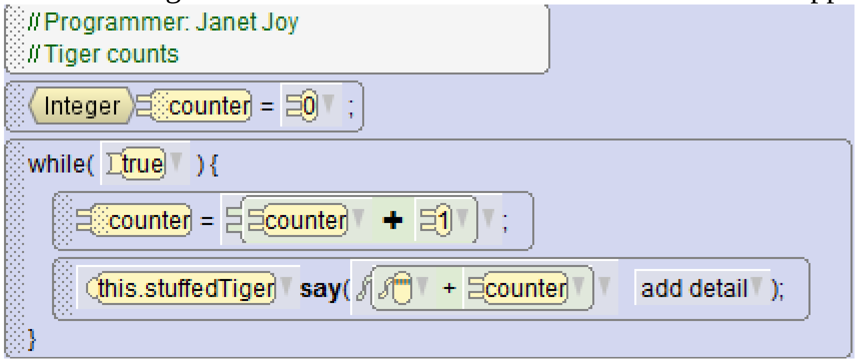

position
this.snowMan getDistanceTo(other: ???
this.snowMan getDistanceAbove(other: ?
this.snowMan isAbove(other: ???)
this.snowMan getDistanceBelow(other: ?
this.snowMan isBelow(other: ???)

Decimal numbers have the same operators, but more functions:


min, max
absolute value, round, ceiling, floor
sqrt, pow
sin, cos, tan, asin, acos, atan, atan2, PI
exp, log, E

Counters

A very common task is to count how many times something happens. To start, declare an integer variable and give it an initial value of 0. Each time the event happens add 1 to the counter.



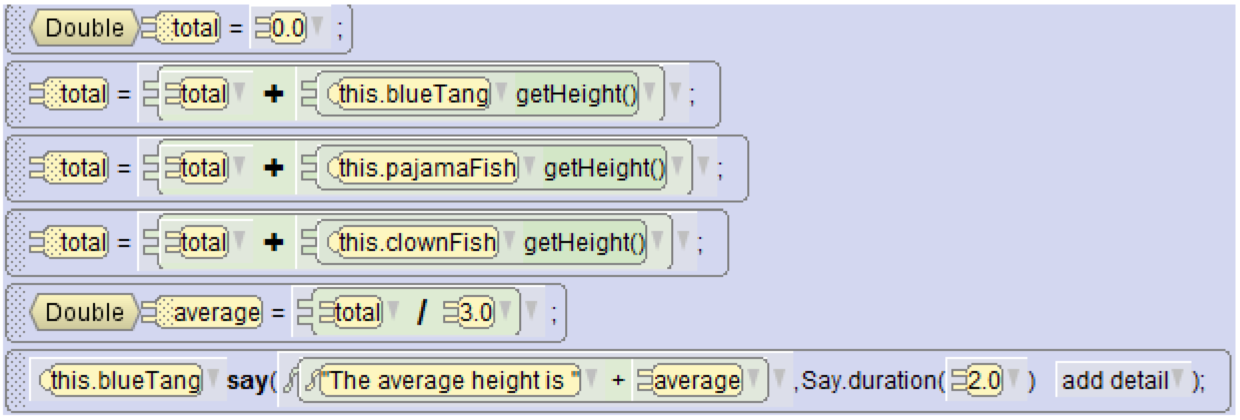
```
//Programmer: Janet Joy
//Tiger counts
Integer counter = 0 ;
while( true ){
    counter = counter + 1 ;
    this.stuffedTiger say( + counter add detail );
}
```

In this example the tiger counts starting with 1. If we switch the order of the statements in the while loop, the tiger will count starting with 0.

Accumulators

Another common task is to find a total of different values. To start, declare a variable and give it an initial value of 0. The type should be the same type as the values you want to add.

In this example there are 3 fish. We want to find the average height. To find the average we first must find the total.



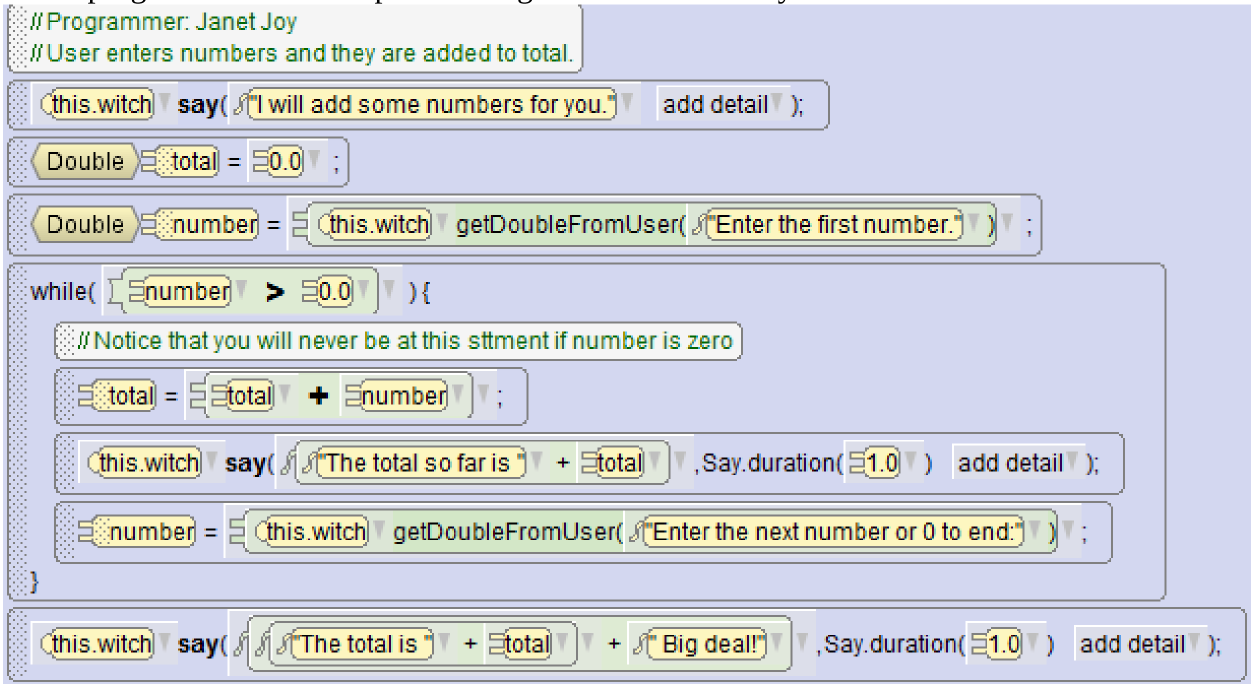
```
Double total = 0.0 ;
total = total + this.blueTang getHeight() ;
total = total + this.pajamaFish getHeight() ;
total = total + this.clownFish getHeight() ;
Double average = total / 3.0 ;
this.blueTang say( "The average height is " + average ,Say.duration( 2.0 ) add detail );
```

Note that we could have used
total=this.blueTang.getHeight()+this.pajamaFish.getHeight()+this.clownFish.getHeight()

However, the method shown of adding each number to total one at a time is easier to understand and it is also easier to add a fourth fish.

A Running Total in a Loop

In this program the witch keeps a running total of the values you enter.



```
// Programmer: Janet Joy
// User enters numbers and they are added to total.
this.witch say( "I will add some numbers for you." add detail );
Double total = 0.0 ;
Double number = this.witch getDoubleFromUser( "Enter the first number." ) ;
while( number > 0.0 ) {
    // Notice that you will never be at this sttment if number is zero
    total = total + number ;
    this.witch say( "The total so far is " + total ,Say.duration( 1.0 ) add detail );
    number = this.witch getDoubleFromUser( "Enter the next number or 0 to end:" ) ;
}
this.witch say( "The total is " + total + " Big deal!" ,Say.duration( 1.0 ) add detail );
```

Notice that the first number is entered before the loop and each subsequent number is entered as the last statement in the loop. The body of the loop is not executed if the number is 0 or less.

Experiment:
Add a counter to this loop and display the average at the end.

Smallest and Largest

One of the advantages of getting the first value before the loop is that you can set smallest and largest to the first number. This modification of the program finds the smallest and largest.

It is shown as Java Code:

```
public void myFirstMethod() {
    //Programmer : Janet Joy
    //User enters numbers and they are added to total.
    this.witch.say( "I will add some numbers for you." );
    Double total = 0.0;
    Integer count = 0;
    Double number = this.witch.getDoubleFromUser( "Enter the first number." )
;
    Double smallest = number;
    Double largest = number;
    while ( number>0.0 ) {
        //Notice that you will never be at this sttment if number is zero
        total = total+number;
        count = count+1;
        if( number<smallest ) {
            smallest = number;
        } else {
        }
        if( number>largest ) {
            largest = number;
        } else {
        }
        this.witch.say( "The total so far is "+total, Say.duration( 1.0 ) );
        number = this.witch.getDoubleFromUser( "Enter the next number or 0 to
 end : " );
    }
    this.witch.say( "The total is "+total+" Big deal!", Say.duration( 1.0 ) )
;
    this.witch.say( "The numbers ranged from "+smallest+" to "+largest );
}
```

You can see the Java code by selecting Window-> Preferences->Java Code on the Side.