Variables

A variable is a *name* that holds a *value*. The first thing you do with a variable is *initialize* it by setting a *name* to any *value*. After that you can use that *name* to edit or use the *value*.

```
x = 5
y = 2
z = x-y # the value of z will be the value of x minus the value of y
```

If-Statements and Conditionals

A condition can evaluate to either **True** or **False**. You can think of a condition as a statement that is either **True** or **False**. An **if-statement** is a condition with a block of code that will only run **if** the **condition** is **True**. If the condition is false then the code in an **else** block will run if there is an **else block**.

```
# conditions read like
# (x == 5)           x is equal to 5
# (x > 5)       x is greater than 5
# (x < 5)       x is less than 5
# (x < 5 & x < 2)     x is less than 5 and x is less than 2
# (x < 5 | x < 2)     x is less than 5 or x is less than 2
# not(x <  2)          x is not less than 2

x = 2
if ( x < 5 ):
     print("x is less than 5")
else:
     print("else")
```

Lists

A **list** in Python is similar to a **list** in real life. All a **list** is, is a collection of values. You find an item in a **list** with it's **index**. An **index** in a Python list would be like a real list's line number. **Indexes** in Python start at 0 so the first item in a Python **list** would be item 0. You can also loop through lists shown in the Loops section. **Lists** use square brackets to define the sides, and commas to separate the values.

```
myList = ["a","b","dog","yes"] # a list named myList
myList.append("hello") # adds "hello" to the end of the list
myList[0] # the first index of myList which is "a"
myList[4] # the fourth index of myList which is "hello"
```

Loops

A loop is used to make a block of code run more than one time. There are two types of **while loops**, and **for loops**. A **while loop** will loop the code inside *while* a **condition** is **True** similar to an if statement. A **for loop** will run *for* a certain amount of *iterations*.

```
# a while loop that runs 5 times
x = 0
while (x < 5):
      # do something
      x = x+1

# a for loop that loops 5 times
# i is the iterator
for i in range(5):
      # do something

myList = ["a","b","dog","cat"]

# loops through each thing in myList
# item will be like a variable that is whatever iteration in the
myList the loop is on
for item in myList:
      # do something to item
```

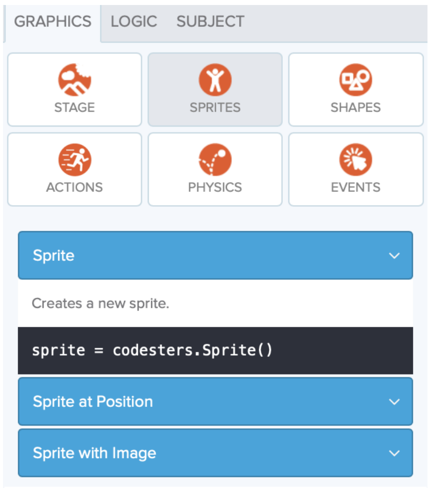
Functions

A **function** is a name for a block of code. A **function** may take input which is called a **parameter**. A **function** may also **return** a value. The block of code that **defines** the function is run anytime the **function** is **called**. The **parameters** that are input to a **function** can be accessed by their name only inside of the **function**. A function can **return** a value which means that when the **function** is called it will be equal to something.

```python
# a function is defined using the def keyword
# this function takes a parameter named myParameter and prints it
when called
def myFunction(myParameter):
      print(myParameter)

# calling myFunction with the "hello" param will print hello
myFunction("hello")

# this function takes 2 parameters and adds them together then
returns the result
def addTwoNumbers(firstNum, secondNum):
      result = firstNum + secondNum
      return result

# calling the add 2 numbers function returns 3+2
finalNumber = addTwoNumbers(3,2)
# will print 5 because addTwoNumbers returned 3+2 which is 5
print(finalNumber)
```

Codesters

Python has **libraries** which are a collection of variables and functions that other people can use. **Codesters** is a python library that has functions that can help us to put things on a screen. You can see examples on the left side of the Codesters editor here. In this example codesters.Sprite() is a function that gives us a new sprite.



Creates a new sprite.

```python
sprite = codesters.Sprite()
```