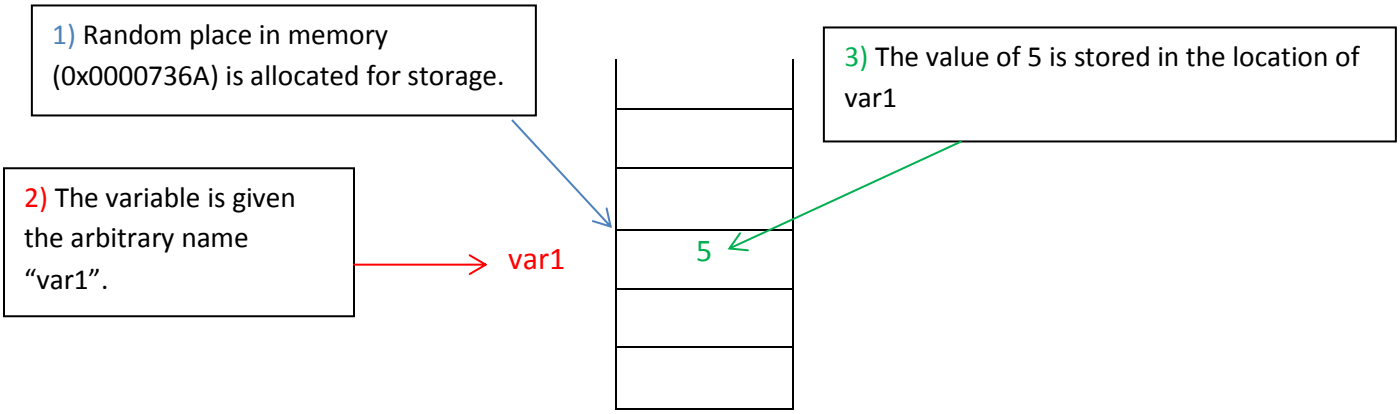
Reference Sheet – 1

Variables

* A variable is a location in memory where information (in the form of a value) is stored.
* Places in memory have addresses (E.g. 0x0000883A).
* To make referencing an address easier, variables are assigned arbitrary names.
* The statement `var1 = 5` is interpreted as: "give me a space in memory, name that space 'var1', and assign the value of 5 to that location. Visually, it looks like:



* Calling the variable "var1" accesses its specific place in memory and the value stored in that location.
* Variables have special types. Common variable types include:
- Integers (int) – examples: 1, 7, 0, 1387, -5
- Doubles (float, floating-point) – examples: 1, -8, 0, 3529, 6.5, 0.345, -7.53
- String – examples: "hello", "world", "hello world", "asdfjkl;!@#$%^&*", "a"
- Bool (Boolean value) – examples: True (true, T), False (false, F)

** Note: When performing operations and comparisons, it's important to pay attention to the type of the variables you're using – you may get errors or unexpected results if you don't.*

Printing

* Programs commonly use print statements to communicate with users.

* In Python 2.X, use: `print "Hello World"` and In Python 3.X, use: `print("Hello World")`

* The print function works with any value or variable (by using its name) regardless of type:

```
secretNumber = 12345

print secretNumber #prints "12345" to the shell
```

Conditional Statements

* Conditional statements depend on the evaluation of Boolean values using comparisons.
* Booleans have a value of either *true* or *false*.
* Comparison operators include:

<	(less than)
>	(greater than)
==	(equal to or is equivalent to)
!=	(not equal to or is not equivalent to)
<=	(less than or equal to)
>=	(greater than or equal to)

* Conditional statements are composed of three parts:

1) Declaration
2) Comparison
3) Block

```
a = 5
b = 3

if a > b:
    print "a is larger"
```

1) The declaration is made by using the statements if, elif, and else -in that order, and only as needed.

2) The comparison is an expression made up of either two values or variables and an operator. The expression must be punctuated with a colon: ":".

3) The block is code that runs *only if* the parent condition is true, otherwise it is ignored. *The block must be indented one tab relative to the parent condition!*

* Conditional statements can become Complex with the addition of elif (short for "else if") and else statements.

```
a = 5
b = 5

if a < b:
    print "a is smaller"
elif a > b:
    print "a is larger"
else:
    print "a is equal to b"
```

The elif statement runs *only if* the parent if statement is false.

The else statement runs *only if* all of its parent conditions are false. The else statement has no expression to evaluate and will run its block automatically under the condition its parent statement(s) are false.

* The easiest way to think about comparisons is to read them as a question. "Is the variable a less than b?" If the answer is yes, then the comparison is true and the block of code following it will run.

User Input

* The user can give a program information using the function `input()` or `raw_input()`.
* The user's input has to be assigned to a variable in the source code to be used.
* `input()` will evaluate the user input as Python code.
* `raw_input()` evaluates the user input as a literal string.
* Using either user input function inappropriately can bug your program.

For example, if the program needs the number 5 to be represented as a string ("5") and not an integer, the return value of `input()` will automatically convert the input to an integer.
On the other hand, if the program needs the number 5 to be represented as a string ("5") and not an integer, using `raw_input()` will correctly return the input as the string "5".

* Both `input()` and `raw_input()` take a string *parameter*, which appears as a prompt on the shell to guide the user.

* Type casting can force variable values to change types. Casting must follow simple logic.
    - `int()` -casts the parameter to become an integer
    - `str()` -casts the parameter to become a string

* An example:

```
#assign the variable 'a' a string version of the number provided by the user
      a = raw_input("Give me a number")

print a #assume the user provided the number 3

print a + 1 #this will cause an error, a string and an integer can't be summed

#convert 'a' to an integer
a = int(a)

print a + 1 #this will print 4
```