15110 PYTHON REFERENCE SHEET

Arithmetic Operations:	`**`	`*`	`/`	`//`	`%`	`+`	`-`
Relational Operations:	`==`	`!=`	`<`	`<=`	`>`	`>=`	
Logical Operations:	`and`	`or`	`not`				

Variable Names: All variable names must start with a letter (lowercase recommended). The remainder of the variable name (if any) can consist of any combination of uppercase letters, lowercase letters, digits and underscores (_). Variables are case sensitive.

Assignment Statement: *variable* = *expression*

Defining a function:

```
def functionname( parameterlist ):
        function_body
```

A *parameterlist* may be empty or may include one or more variables representing data required for the function, separated by commas.

Calling a function: *functionname* (*argumentlist*)

An *argumentlist* may be empty or may include one or more expressions representing data required for the function to use, separated by commas.

Importing module: `import` *modulename*

Using module: *modulename* . *functionname* (*argumentlist*)

`print(data)`	prints data to screen and moves cursor to next line
`print(data,end=" ")`	prints data to screen and keeps cursor on same line
`print()`	moves cursor to next line
`return(data)`	returns data to instruction that called this function

```
for v in range(x, y, z):          loops for v = x through y-1, inclusive in steps of z
        loop_body                 (y is optional, default 0.   z is optional, default 1.)

while  condition:                 loops while condition is True
        loop_body

if  condition1:                   executes instruction1 set once if condition1 is True
        instruction1_set
elif  condition2:                 otherwise executes instruction2 set once if condition2
        instruction2_set                  is True. This part is optional, can be repeated.
else:                             otherwise executes instruction3 set once if all
        instruction3_set                  previous conditions tested as False. Optional.
```

Lists:	*listname* = `[]`	An empty list.
	listname = `[` $item_0$, $item_1$, … , $item_{n-1}$ `]`	A list of n items, n >= 1.
	listname `[` *i* `]`	Evaluates to the ith element of the list

`len(`*listname*`)`	returns the number of items in the list
item `in` *listname*	returns `True` if the item is in the list, `False` otherwise.
listname `[` *i* : *j* `]`	returns a sublist of list from index *i* to *j*-1
listname = `[` *item* `] *` *n*	creates a list with *n* copies of the item
listname `.append(`*item*`)`	appends item to end of the list
listname `.remove(`*item*`)`	removes the first occurrence of the item in the list

```
for item in listname:             performs instructions once for each item in list, no index is available
        loop_body                         (item can be referenced in loop body)
```

Using Random Integers:

```
from random import seed, randint
```

`seed(`*value*`)`	seeds the random number generator using the given integer *value*
`randint(`*x, y*`)`	returns a random integer "uniformly distributed" between *x* and *y*, inclusive

Using the Canvas (graphics):

from PythonLabs.Canvas import *

`Canvas.init(`*width, height, title*`)`
: opens a window of *width* and *height* in pixels with the given *title* (no spaces).

`Canvas.Rectangle(`*x0, y0, x1, y1, optional_parameters*`)`
: draws a rectangle from top left *(x0, y0)* to bottom right *(x1,y1)* in units of pixels.
Optional parameters (separated by commas if using more than one):
`fill = "`*color*`"`
`outline = "`*color*`"`
`width = ` *numpixels*
color may be specified as a name or a hex code (e.g. "blue" or "#0000FF")

`Canvas.Circle(`*x0, y0, radius, optional_parameters*`)`
: draws a circle with center at coordinate *(x0, y0)* and the given *radius* in pixels.
optional_parameters: see list above

`Canvas.Polygon(`*point_list, optional_parameters*`)`
: draws a polygon with vertices taken from the list of points *[x0, y0, x1, y1, ...]*,
as follows: *(x0, y0)*. *(x1, y1)*. ... wrapping around back to *(x0, y0)*.
optional_parameters: see list above

`Canvas.Line(`*x0, y0, x1, y1, optional_parameters*`)`
: draws a line from top left *(x0, y0)* to bottom right *(x1,y1)* in units of pixels
optional_parameters: see list above

`Canvas.Text(`*string, x0, y0,* `anchor = "`*location*`", fill = "`*color*`")`
: Draws the text in the given *string* at *(x0, y0)* given in pixels.
Text is *anchor*ed ("left", "center" or "right") with the given fill *color*.