**Python Student Manual**

**Education Unlimited Summer 2017**

**Command Line Basics:**

Open your terminal. Here you can type commands. Commands can have parameters, which are optional letters you type after the command after a dash (-). Press “Tab” to complete the time of what you’re typing. “.” is the current folder, “..” is the parent folder, “\*” is all files in your location. Common commands:

* ls: list all files in the current location
* cd: change directories.
  + Example: if you have a folder called “stuff” in your location, typing “cd stuff” will move you into the folder “stuff.”
  + Typing “cd ..” will take you up to the parent folder
* mkdir folderName: makes a new folder with the name “folderName”
* rmdir folderName: removes the folder called “folderName”
* rm fileName: removes the file called “fileName”
  + Typing “rm \*” will remove all the files in your current directory
* nano fileName: very simple file creator/editor. Only supports typing. When done, type “control-x”, then “y”.
* clear: clears the screen
* python fileName.py: runs the python file

**Math Operations**

* The plus sign (+) will add integers or doubles. If used with Strings, it will concatenate (merge) the Strings together.
* The minus sign (-) will subtract one number from another.
* An asterisk (\*) is used for multiplication. The order of operations (multiplication, division, addition, subtraction, left to right) is used by Python.
* A slash (/) is used for division. If both numbers are integers, this will return an integer, so 3/2 = 1, not 1.5.

**Variables**

Most often used types of variables:

* Int: an integer
* Float: a floating point value (ex. 1.25)
* Boolean: can only be True or False
* String: a string of characters. Always defined using quotation marks.

Creating a variable:

* name = value

Names can be words and they always start with lowercase letter. The convention for multiword variable names is “first\_second\_third.”

Example of creation and use of variables:

c=1 #integer

isSummer=True #boolean

morning = "Good morning, class!" #String

d = c+1

print "isSummer = ", isSummer

**Math Shortcuts**

* += : This sets the variable equal to itself plus the following number. Ex. i += 4 is the same as i = i + 4.
* -= : This sets the variable equal to itself minus the following number. Ex. i -= 2 is the same as i = i - 2.
* \*= : This sets the variable equal to itself multiplied by the following number. Ex. i \*= 5 is the same as i = i \* 5.
* /= : This sets the variable equal to itself divided by the following number. Ex. i /= 3 is the same as i = i /3.

**Logic Operators for Numbers**

* == : Equals. Returns true if the two inputs are equal, and false if the two input are not equal.
  + 4==4 returns true, “hello”== “hi” returns false.
* != : Not Equals. Returns true if the two numbers are not equal, and false if the two numbers are equal.
  + 6!=7 returns true, “ciao”!= “ciao” returns false.
* > : Greater than. Returns true if the first number is greater than the second, and false if it is not. For Strings, returns true if the first word comes after the second alphabetically.
  + 7>4 returns true, 7>10 returns false, 7>7 returns false.
  + “apple”> “zebra” returns false, “tree” > “bobcat” returns true, “lake”>“lake” returns false.
* < : Less than. Returns true if the first number is less than the second, and false if it is not. For Strings, returns true if the first word before after the second alphabetically.
  + 3<8 returns true, 3<1 returns false, 3<3 returns false.
  + “apple”<“zebra” returns true, “tree” <“bobcat” returns false, “lake”<“lake” returns false.
* >= : Greater than or equal to. Returns true if the first number is greater than or equal to the second, and false if it is not. For Strings, returns true if the first word comes after the second alphabetically or if they are equal.
  + 9>=1 returns true, 9>=9 returns true, 9>=12 returns false.
  + “apple”>= “zebra” returns false, “tree” >= “bobcat” returns true, “lake”>=“lake” returns true.
* <= : Less than or equal to. Returns true if the first number is less than or equal to the second, and false if it is not. For Strings, returns true if the first word comes before the second alphabetically or if they are equal.
  + 2<=5 returns true, 2<=2 returns true, 2<=0 returns false.
  + “apple”<=“zebra” returns true, “tree” <=“bobcat” returns false, “lake”<=“lake” returns true.

**Compound Boolean Logic**

* not : “not” operator. Takes one boolean and reverses it. Returns true if the input is false, returns false if the input is true.
* & : “and” operator. Takes two booleans and returns true if they are both true, and false if either one or both of them is false.
* | : “or” operator. Takes two booleans and returns true if either of them are true, and false if both of them are false.

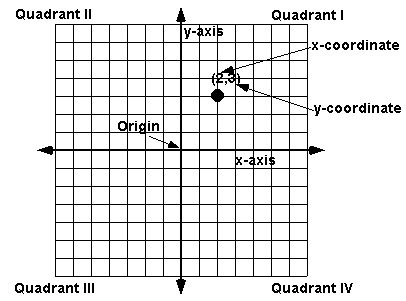
**Capturing User Input**

* Example: name = str(raw\_input("What's your name? "))
* Name is the new variable above.
* You can cast the input to be a specific type. This is often necessary if you’re going to be printing it or doing math with it later.
  + str(raw\_input()) casts the user input as a String
  + int(raw\_input()) would cast the input as an integer
  + float(raw\_input()) would cast the input as a float
* Raw\_input captures the next thing the user types
* “What’s your name? ” prompts the user. If you want to prompt the user in a new line, just use raw\_input().

**Math Concepts for Turtle Graphics**

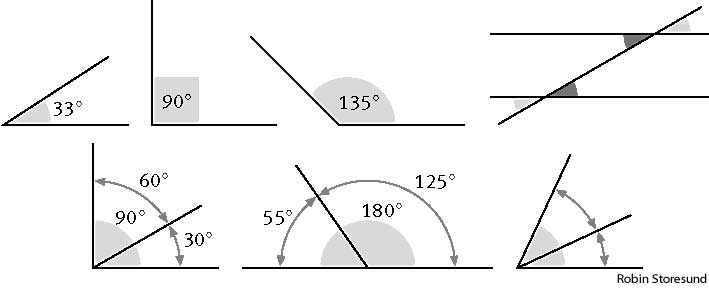
**Graphing**

Points are in (x,y) form. The origin is at point (0,0). To graph a point, first find the x value on the x-axis, with positive numbers to the right and negative numbers to the left. Then, find the y value on the y-axis, with positive numbers up, and negative numbers down. Put a dot at the point where the 2 lines intersect.



**Angles and Degrees**

A degree, usually denoted by **°** , is a measurement of an angle, defined so that a full rotation is 360 degrees. Angles less than 90° are acute, angles equal to 90° are right angles, and angles greater than 90° are obtuse angles. A 180° angle is a straight line. A 360° angles goes all the way around a circle.



**Turtle**

* Turtle is the Python Graphics module.
* To use Turtle, type ‘import turtle’ at the top of your program.
* To create a new Turtle, use the turtle.Turtle() method.
  + Example: bob = turtle.Turtle()
* To change the shape of your turtle, use the shape method. The possile shapes are: “arrow”, “turtle”, “circle”, “square”, “triangle”, “classic”.
  + Example: bob.shape("turtle")
* To move the turtle, use the forward method. Inside, put the number of steps you want the turtle to take.
  + Example: bob.forward(50)
* To make the turtle go to a specific point on the screen, use goto(x, y), setpos(x, y), or setposition(x,y), they all move the turtle to that point.
  + Example: bob.goto(0,0) will put the bob turtle in the middle of the screen.
* To rotate the direction the turtle is facing, use the left() and right() methods. Inside, put the number of degrees you want the turtle to turn by.
  + Example: bob.left(90)
* To change the color that your turtle is drawing in, use the color() method. Possibilities include all of the regular “named” colors
  + Example: bob.color("blue")
  + For custom colors, go to this site: <https://www.webpagefx.com/web-design/color-picker/> and use the 6-digit code for the color of your choice.
  + Example: bob.color(“#90C3D4”)
* To change the width of the line the turtle draws, use the pensize() method.
  + Example: bob.pensize(3)
* To allow the turtle to move without drawing, use the penup() method.
* To return to drawing when the turtle moves, use the pendown() method.
* The turtle window automatically closed when all the lines of code have executed. However, to have the window wait to close until you click on it, use the exitonclick() method.
  + Example: turtle.exitonclick()
* To have the turtle window close without a click, use the bye() method.

**Conditionals (If-Else Statements):**

* Conditionals allow certain statements in Python to only run under certain conditions.
* If Statements consist of the word ‘if’ followed by a boolean expression (something that evaluates to true or false, usually in parentheses). If the expression evaluates to true, the following code that is indented below will execute. If the boolean expression is false, then the code with be skipped.
  + Example:

if (age >= 10):

print "Congrats! You hit double digits!"

* + If you enter 7, then nothing will print. If you enter 11, then "Congrats! You hit double digits!" will print.
* If-Else statements have an If statement followed by an Else statement. The code block after the if will execute if the boolean expression is true, and the code block after the else will execute if the boolean expression is false.
  + Example:

if (age >= 10):

print "Congrats! You hit double digits!"

else:

print “You’re still a kid!”

* + If you enter 7, then “You’re still a kid!” will print. If you enter 11, then "Congrats! You hit double digits!" will print.
* Optionally, there can be ‘Else if’(elif) statements between the if and the else statements. In a string of these statements, only 1 will ever run.
  + Example:

if (age >= 10):

print "Congrats! You hit double digits!"

elif (age > 3):

print “You’re a kid!”

else:

print “You’re a baby!”

* + If you enter 7, then “You’re a kid!” will print. If you enter 11, then "Congrats! You hit double digits!" will print. If you enter 2, then “You’re a baby!” will print.

**While Loops**

* While loops are often used to keep doing a process until something happens - until a user enters the magic password, rolls doubles, etc. Sometimes they are used to do something a set number of times using a counter.
* While loops consist of the word ‘while’ followed by a boolean expression (something that evaluates to true or false, usually in parentheses). If the expression evaluates to true, the following code that is indented will execute and then it will check the boolean expression again. As long as the expression is true, the loop will keep repeating. If the boolean expression is false, then the code with be skipped and execution will move on to the next code outside of the while loop.
* Warning: Infinite loops! If you do not ever change the boolean expression and it stays true, your program will go into an infinite loop and you will have to manually stop it. Make sure there is always something in your while loop that will eventually cause the boolean expression to become false.
* Example:

i=0

while (i<5):

print i

i+=1

This will print the numbers 0 through 4 on separate lines.

**Random Numbers**

To generate a random number, type ‘import random’ at the top of your project. Then, for whole numbers, use random.randint(min, max). For example, to generate a random whole number between 1 and 50, use:

rand = random.randint(1,50)

To choose a random element from a list (lists are below)

**Lists**

A list is a collection of elements that may have different types. The length of the list (the number of things in it) changes as you add and remove elements. The square brackets [ ] are used to represent a list. The indices (singular index) for lists start at 0, so if you have a list of size 10, the indices go from 0 to 9.

* Initializing lists:
  + To initiazlie an empty list: my\_list = []
  + To initiate a list with values (can contain mixed types): my\_list = [1, "yay", 140.76]
* Finding information about a list:
  + To get an item from a list by index: my\_list= [1, 2, 3, 4, 5, 6]
    - my\_list[0] will give the first item in the list, which is 1
    - my\_list[3] will give the 4th item in the list, which is 4
  + To get the length: length = len(my\_list)
  + max(my\_list) will return the largest element of the list
  + min(my\_list) will return the smallest element of the list
  + (3 in my\_list) will return a boolean: true if 3 is in the list, false if it is not in the list.
* Adding to a list:
  + To add something at the end of the list: my\_list.append(8)
  + To insert something at a certain index in the list: my\_list.insert(6, 7), which will insert 7 at index 6. So the usage is insert(index, object).
  + To change a value in a list: my\_list[3] = 12 will set the 4th item to 12.
* Removing from a list:
  + To delete something from a list: del my\_list[2] will delete the 3rd item.
  + To delete a specific object from the list: my\_list.remove(5) will remove all the 5s in the list.

**For Loops**

* For loops are used to do something a certain number of times or to go through an array and do something to each element.
* For loops consist of the word ‘for’ followed by a variable which will represent each element in the sequence, then the word ‘in’, then the name of the sequence (often a list or range), then a colon(:). Below, indented, is the code that will be repeated. The general form is:

for <variable> in <sequence>:

print <variable>

* Example: If you have a list called my\_list, this will print all the elements of my\_list on separate lines.

for i in my\_list:

print i

* Using range in a for loop: if you don’t have a list, but you want to repeat something a fixed number of times, use range(n) to generate a list of integers from 0 to n-1. For example, range(10) = 0,1,2,3,4,5,6,7,8,9. You can also use range(begin, end) if you don’t want to start at 0. This will generate a list of integers from begin to end-1. For example, range (3,7) = 3,4,5,6.
* Example: The following will print “hello” 3 times

for x in range(3):

print “hello”

**Functions**

Functions are sets of code which have a name and can be called (used) at any point in a program by using its name. Methods are like subprograms that act on data and often return a value.

* Example of a void function with no input variables:

def sayHello():

print “Hello!”

return

* Example of a function which takes in 2 numbers and returns their sum:

def addNums(x,y):

return x+y