Python Quick Start

Python is an interpreted language and easy to learn. Right now rising most quickly in terms of popularity. Python 2 and 3 are both in use, mainly python 2 is still around because there are a lot of libraries and modules written already for it but 3 has been out for a while and is the future of python.

Free web references for Python:

https://automatetheboringstuff.com/ Intro level

http://www.diveintopython3.net/ More advanced, covers closures, iterators

http://interactivepython.org/runestone/static/pythonds/index.html Focus on Data Structures

You can run programs interactively or from an IDE or from the command line.

Lots of web resources "learn python" where you can type programs in on a webpage and run them.

```
print("hello")
```

Indentation matters, not like Java or Matlab

```
x = 1
if (x==1):
    print("x is 1")
```

You can make floating point and integer numbers. You don't need to declare your variables, you can just start using them.

To comment something out use # in front

Strings are in " or '

Variables change type based on what they were last assigned to. Use int(var) or float(var) to convert types.

You can print multiple things with a comma. Don't try to use + to concatenate strings and numbers.

```
print("hello",x)
```

You can also use the formatted output:

```
z = 4.599
s = "Bob"
print("My number is %.2f and name is %s" % (z,s))
```

You have your normal mathematical operators: +, -, *, /, %

If you want integer division, use // and use ** for exponent

If statement:

We previously saw this is:

```
if (boolean):
       code
elif (boolean):
       code
else:
       code
```

Relational operators that you can use are ==, != <, >, <=, >=

To input, use `var = input("prompt")`

```
while (bool):
else:

for x in range(1,5):              #1-4
```

Python has a cool built in list type. If you have ever been frustrated dealing with static arrays, then you will be pleased with lists. You can build them up on the fly, tear them down on the fly, access them like arrays, and other things.

```
list1 = []
list2 = [1, 2, 3.5, "hello"]
print(list2[0])            # 1
print(list2[2])            # 3.5
print(len(list2))          # 4   length of list
```

You can add on to a list with append:

```
list1.append(99)
list1.append(100)          # [99, 100]
```

Replace items with something else:

```
list2[1] = 999             # [1, 999, 3.5, "hello"]
```

Use remove to remove an element from the list:

```
list1.remove(99)           # remove item 99
```

Pop will take off the last item from the list, or from a specified index:

```
list2 = [1, 2, 3, 4]
x = list2.pop()            # x = 4,   list2 = [1,2,3]
x = list2.pop(1)           # x = 2,   list2 = [1, 3]
```

Lists can contain lists!

```
phoneBook = [["Bob","786-1234"],["Ted","786-1111"],["Alice","999-9999"]]
```

You can loop over lists with the for loop

```
name = input("Enter a name")
for n in phonebook:
	if (n[0] == name):
		print(n[1])
```

python will even search lists for you:

```
list2 = [1, 2, 3, 4]
3 in list2                              # true
12 in list2                             # false
list2.index(2)                          # 1
```

You can join two lists with +

```
list1 = [1, 2, 3, 4]
list2 = [10,11]
list1+list2                             # [1,2,3,4,10,11]
```

Lists are accessed by reference. This means that if you assign one list to another, you really have two list variables that are referencing the same thing.

```
list1 = [1, 2, 3, 4]
list2 = [10, 11]
list2 = list1
print(list2)                            # [1,2,3,4]
```

Functions in python allow us to divide our code into blocks, or procedures. This is basically the same thing as a method in Java or function in C++. The format for a function looks like this:

```
def function_name(arguments):
	code for function
	return value
```

Here are some simple examples:

```
def printNums():
	for x in range(1,11):
		print(x)

def findPhone(phones, target):
	for n in phones:
		if (n[0] == target):
```

```
                        return(n[1])

    printNums()
    phoneBook = [["Bob","786-1234"],["Ted","786-1111"],["Alice","999-9999"]]
    p = findPhone(phoneBook,”Ted”)
    print(p)
```

In python, essentially every variable is an object. Variables are passed by “object reference”. For numbers and strings, you get the same behavior as call by value. Changes made in a function to a variable don’t affect it outside the function. For things like lists, where you can change the innards of the object, the changes are seen outside the function.

```
    def foo(x, l):
        x = 100
        l.append(“hi”)

    x=1
    l=[1,2]
    foo(x,l)
    print(x)                    # still 1, not changed to 100
    print(l)                    # now contains [1,2,’hi’]
```

Here is a Simon Game

```
import msvcrt
import random
import time
import winsound

def playSound(dir):
    if (dir=="left"):
        winsound.PlaySound("slide_whistle_down.wav",winsound.SND_FILENAME)
    elif (dir=="right"):
        winsound.PlaySound("slide_whistle_up.wav",winsound.SND_FILENAME)
    elif (dir=="down"):
        winsound.PlaySound("s500hz.wav",winsound.SND_FILENAME)
    elif (dir=="up"):
        winsound.PlaySound("s1000hz.wav",winsound.SND_FILENAME)
    else:
        winsound.PlaySound("burp.wav",winsound.SND_FILENAME)

def addDirection(seq):
    val = random.randrange(1,5)   #1-4
    if (val==1):
        d = "up"
    elif (val==2):
        d = "down"
    elif (val==3):
        d = "left"
    elif (val==4):
        d = "right"
    seq.append(d)
```

```
def showSequence(seq):
    print("Next sequence!")
    for d in seq:
        print(d,end=" ")  # space after print statement instead of newline
        playSound(d)
        time.sleep(0.5)
    print()

def inputSequence(seq):
    print("Input sequence");
    for dir in seq:
        key = ""
        while msvcrt.kbhit():
            msvcrt.getch()
        ch = ord(msvcrt.getch())
        if ((ch == 224) or (ch==0)):  # special keys like arrow
            keychar = ord(msvcrt.getch())
            if keychar == 80:
                key = "down"
            elif keychar == 72:
                key = "up"
            elif keychar == 75:
                key = "left"
            elif keychar == 77:
                key = "right"
        playSound(key)
        if (dir != key):
            return True
    return False

def main():
    seq = []
    gameover = False
    while (not gameover):
        addDirection(seq)
        showSequence(seq) # Comment out to make game challenging
        gameover = inputSequence(seq)
        time.sleep(1)
    print("Game Over!")

# Calls main function
main()
```