Python 2: Loops & Data Input and output

Python function refresher

There are many python functions. We will use a few in this section. Reminder that documentation is available here.

https://docs.python.org/3/library/functions.html

Logic in program flow

Need to be able to specify logic

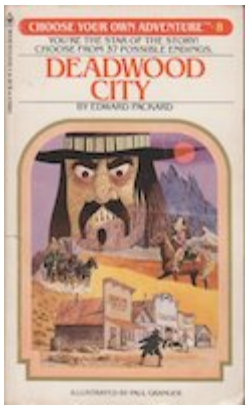
```
if TRUE:
    THEN DO X
else
    THEN DO Y
```

And Loop

```
while ( TRUE ):
    Do X
```

Python uses indentation

- Python uses indentation to indicate the branching in the logic.



This indentation can be either spaces or tabs. Emacs/Atom/Vi allow customizing your editor so that when [tab] is typed it will either insert 4 spaces or a tab. Switching between editors while editing a file can sometimes cause problems.

Solution: delete the indendentation and enter the tab (or spaces) again.

After this slide you will now understand this scene from Silicon Valley.

Logical operators

- === for equality. Returns true if two values are equal. Do not use = which is assignment

```
x=10
if x == 10:
print("this is a ten")
```

- != for not equal.
- < - less than > - greater than <= less than or equal to >= greater than or equal to

```
a = 7
b = 82
if b > a:
print("b is greater than a")
```

Other operators

- `is` - tests for the content of the string being the same

```
a="fiat"
b="".join(['f','i','a','t'])
if a is b:
   print("strings are same")
if a == b:
   print("these are the same string: %s %s"%(a,b))
```

Logical operators to combine or modify statements

- Not - not
- Or - or
- And - and

```
a=10
b=12
c=13
if a < b and b < c:
    print("a is smaller than c")
```

```
if not a == b:
    print("a is not equal to b")
else:
   print("a is equal to b")
```

Loops

- while loops run as long as condition is true

```
x=1
while( x < 10 ):
print("x is",x)
x += 1
```

- for loops loop through a set of items

```
DNA="AAAGCCAAG"
for base in DNA:
print("base is %s"%(base))
```

Python range operator

Simple way to setup a counter. See the range function

```
for i in range(1,10): # forwards counting
    print(i)
for i in range(10,0,-1): #backwards counting
    print(i)
for i in range(2,16,2): # count by twos
    print(i)
```

Loop on a list

Iterate on a list using indexes

```
list = (7, 10, 2, 2, 7)
for i in range(len(list):
    print("list[%d] = %s"%(i,list[i]))
```

Iterate on a list directly

```
for item in list:
   print("item is",item)
```

Exiting the loop

- the `break` function stops a loop and exits it
- the `continue` stops executing in the loop and starts back at top

```
list = ('a', 'b', 'c', 'd')
for l in list:
    if l == "b":
        continue
    print(l)
```

```
a
c
d
```

File handles / Data streams

The open function is used to open file handles. Good reference can be found at https://en.wikibooks.org/wiki/Python_Programming/Input_and_Output

Data streams could be from cmdline (eg STDIN)

```
$ cat file | python myscript.py
```

Can also be a file

```
filehandle = open(myfile,"r")
```

Getting data into your program

`open` - is for opening a file

There are two arguments, one is the filename, the other is how to open, reading, writing. For text data use "r" but if binary data use "rb".

```
file = "data1.dat"
fh = open(file,"r")
for line in fh:
    print(line)
```

Alternative structure for opening

This will throw a warning if the file cannot be opened

```
with open(myfile,"r") as fh:
    for line in fh:
        print(line)
```

Writing data

Write data or text to a file.

```
ofh = open("my_data.tab","w")
ofh.write("Species\tHabitat\tSize\n")
ofh.write("Crab\tBeach\tM\n")
ofh.write("Fish\tOcean\tS\n")
```

```
$ cat my_data.tab
Species    Habitat    Size
Crab    Beach    M
Fish    Ocean    S
```

Modules/Libraries/Packages

Modules are collections of code routines. We will talk more about functions/routines in next lecture. Can use these as tools.

- sys - System-specific parameters and functions
- urllib.request - URLs for opening web or network connections
- csv - Comma and Tab delimited data parsing

Reading in STDIN

Remember we can pass data into a program via STDIN if we use the '|' "pipes" in UNIX

```
import sys

counter = 0
for line in sys.stdin:
    counter += 1
print ("There are",counter, "lines")
```

Data streams don't have to be files

Can be a network connection (eg URL for web or FTP)

```
import urllib.request
orginfo = "https://biodataprog.github.io/programming-intro/data/random_exon
info = urllib.request.urlopen(orginfo)
for line in info:
    linestrip = line.decode('UTF-8').strip()
    print(linestrip)
```

CSV module

```
import csv

file2 = "test2.csv"
with open(file2) as csvfile:
    reader = csv.reader(csvfile,delimiter=",",quotechar='|')
    for row in reader:
        print("\t".join(row))

with open("outtest.csv","w") as csvfile:
    writer = csv.writer(csvfile,delimiter="\t")
    writer.writerow(["Name","Flavor","Color"])
    writer.writerow(["Apple","Sweet","Red"])
    writer.writerow(["Pretzel","Salty","Brown"])
```

command line arguments

- import sys
- get the command line input as a list from sys

```
$ python argparse.py arg1 arg2 arg3 this-is-one "this is one"
```

```
import sys
for n in range(len(sys.argv)):
    printf 'argv[%d] = %s' %(n, sys.argv[n])
```

```
argv[0] = argparse.py
argv[1] = arg1
argv[2] = arg2
argv[3] = arg3
argv[4] = this-is-one
argv[5] = this is one
```