

Python Programming

Flow Control

Outline - Flow Control

- `if`, `elif`, and `else`
- `for` loops
- `while` loops
- Putting it all together, writing a program

Flow Control

- Controlling the flow of a program's execution:
 - The heart of programming
- It means:
 - Specifying that only certain parts of a program will run, or that some parts will run repeatedly, according to conditions, such as value of variables.

Examples

If variable X is bigger than Y, do one thing. If not, do something else.

While the value of $X < Y$, do something, and add 1 to X. If not, do something else.

If function X executes without an error, do something. If not, do something else.

Truth Testing

- If expression evaluates to true, do one thing. If not, do something else:

```
if expr1:  
    expr2  
else:  
    expr3
```

```
if x > y:  
    x = x - 1  
else:  
    x = y
```

The "else" part is optional

Three ways to use “if”

- if, no else
- if, with else
- if, with elif and else

If and no else


```
A = 5
if A>5:
    print "A is > 5"
A = 2
print "Now A is 2"
print "Continuing..."
```



And continue with the program!

If and else


```
A = 5
if A>5:
    print "A is > 5"
else:
    print "A is < 5"
print "Continuing"
...
```



And continue with the program!

If with elif

```
A = 5
if A>5:
    print "A is > 5"
elif A<3:
    print "A is < 3"
else:
    print "A is just right!"
print "Continuing"
...
```



And continue with the program!

Use `elif` to test several conditions

```
def test(value):  
    """  
    Function: Find out if a value is in range  
    Returns : True if yes, False if not  
    Args    : value - the value to test  
    """  
    if value > 50:  
        print "That's too much."  
    elif value < 40:  
        print "That's not enough."  
    else:  
        print "That's just right."
```

You can include many `elif`'s

Doc string

Loops - while

- The body of the `while` loop executes over and over until `testexpr` stops being true.

`while testexpr:`

`expr`
`expr`
`.. (and more)`

This is the body of the while loop

`expr`

Loops - while

- The body of the while loop executes until variable I gets too big
- Watch out for infinite loops!
- Use CTRL-C to escape an infinite loop.

```
A = [1,2,3]
i = 0
while i < len(A):
    print "okay!"
    i = i + 1
print "Done!"
```

Ex) reading a file with - while

- The body of the while loop executes until the condition is no longer true.
- Since 1 is always true, we continue looping until the if statement evaluates to True.
- This happens when we run out of lines to examine.
- The break command sends us to the next statement beyond the loop.

```
def get_mRNA_accessions(fn=None):
    """
    Function: Retrieve a list of all mRNA accessions in the file
    Returns : a list of mRNA accessions (none are duplicated)
    Args    : fn - the name of the file to parse.

    fn - is in bed format.
    """
    d = {} # for temporary storage
    fh = readfile(fn)
    while 1: # this always evaluates to true
        line = fh.readline() #
        if not line:
            fh.close() # the while loop terminates when
            break      # there are no more lines to read
        txt = line.rstrip() # remove trailing whitespace
        values = txt.split('\t')
        accession = values[3]
        if d.has_key(accession):
            continue # we already saw it
        else:
            d[accession]=1
    return d.keys()
```

for loops

- Syntax:
for *variable* in *sequence*:
 block of statements
- Often you can replace with a list comprehension
- But usually not – often you need to do a lot of complex operations within the body of the for loop.

for loops

- Example:
 - For each item in the list, do something

```
for v in range(1,10):  
    print str(v)  
print "v is now: " + str(v)
```

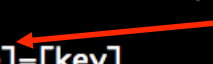
Controlling the flow in a loop

- To exit the loop, use `break`
- To continue without doing anything, use `continue`

EXAMPLE `for` loops

```
def reverse_dict(d):  
    """  
    Function: Make a new dictionary from d where values  
              become keys and keys become values.  
    Returns : a dictionary  
    Args    : d - a dictionary  
    """  
    new_d = {}  
    for key in d.keys():  
        value = d[key]  
        if new_d.has_key(value):  
            new_d[value].append(key)  
        else:  
            new_d[value]=[key]  
    return new_d
```


Test your understanding:
What error might this generate?



EXAMPLE for loops

```
def count_dups(d=None, N=2):
    """
    Function: Count how many mRNAs map to multiple locations
    Returns : an integer
    Args    : d - a dictionary, keys are mRNA
               accessions, value are lists representing
               alignments (output from get_mappings)
               N - number of mappings; only count accessions
               with N or more mappings [default is 2]
    """
    total = 0
    for accession in d.keys():
        if len(d[accession]) >= N:
            total = total + 1
        else:
            continue # just keep going
    return total
```

Test your understanding : re-write as a list comprehension



Homework

Write two functions.

Read the homework and get started.

Bring questions to class on Wed.

On Friday, you'll learn how to use subversion, part of the subversion lab involves checking in your program.

The program (final version) is due the week after.

It's not as easy as it looks so GET STARTED NOW!

BONUS SLIDE

Software Development Process

- Analyze the problem
- Determine specifications
- Create a design
- Implement the design
- Test/debug the program
- Maintain the program
- From Zelle textbook, 2.1, 2.2