Python loops and control statements

Python - Iteration and selection constructs

Python provides both iteration and selection constructs for looping and control branching purposes

As part of iteration, it provides while statement and for statement and as part of selection, it provides if statement.

Iteration:

While statement:

The while statement repeats a set of code until a condition is met.

while : set of statement

```
counter = 1
while counter < 5:
    print "Number " + str(counter)
    counter += 1
```

In the above example, the statement print and incrementing the counter variable is repeated until the value of counter is less than 5.

Try:

Write a program to guess a number until the "0" is typed. You should make use of the random module to generate random numbers between 1 and 6.

for statement:

for statement is another iterative construct that iterates for a certain number of times

1. for a in \<sequence>:

   This loop iterates each element of the sequence and assigns it to the variable as it ierates.

```
>>> name = "tanigai"
>>> for n in name:
...    print n
...
t
a
n
i
```

```
g
a
i
```

2. for a in range(1,10): range function will return a sequence object representing a sequence mentioned as argument to the function.

Control statement

- continue
- break
- pass

When to use while and for construct. - FOR construct is used when the iteration is definitive. Eg. loop over the length of the string - "Definitive iteration.

```
-WHILE construct is used when the iteration is not definitive. Guess the name of the person. "In definit
```

Selection statement:

Selection statement selects a path based on the outcome of a comparison.

if .. else statement if statement branches to a set of statement based on the outcome of the comparison. If the outcome is true, the statement subsequence to the if statement gets executed,

if the comparison results in a false, the control is transferred to the else part

Chained condition:

if \<comparison>: elif: else:

For chained selection, use the elif statement

```
if color == "green":
    print "Color selected is " + color
elif color == "red":
    print "Color selected is " + color
elif color == "green":
    print "Color selected is " + color
else:
    print "No color selected"
```

Nested condition:

if \<comparison>: else: if <>: else:

Comprehension:

Comprehension is a way of representing the mathematical notation in python like { x2 | x ∈ ℕ }. It is a concise way of creating sequences. Comprehensions can be created for list,set and generators.

List comprehension:

List comprehension provides a concise way of creation list.It contains an expression, followed by a for statement and then zero or more for/if statements.

The result will be new list formed by evaluating the expression

```
[ expression for item in list if conditional ]
```

Let us consider the logic to identify the even number from 1 t0 10

```
l = []
for i in range (0,10)
    if i%2:
        l.append(i)
```

This can be achieved in a single line using list comprehension as represented below

```
l = [i for i in range(0,10) if i%2]
```

Dissecting list comprehension: *result* = [*transform iteration filter*]

Using functions in list comprehension

```
>> def triple(x):
return x**3

>>> triple_list = [triple(i) for i in range(10)]
>>> triple_list
[0, 1, 8, 27, 64, 125, 216, 343, 512, 729]
>>>
>>>
```

Set comprehensions:

Set comprehension is same as that of list except for representing the comprehension in set notation.

```
>>> {i for i in range(0,10) if i%2}
set([1, 3, 9, 5, 7])
```

Working examples:

- write a program to check is a References:

http://www.pythonforbeginners.com/basics/list-comprehensions-in-python

http://www.openbookproject.net/books/bpp4awd/ch04.html