Introduction to Python

Loops

Topics

1) Conditionals
   a) if, if-if, if-elif, if-elif-else
   b) Ternary operator
2) For Loops
3) While Loops
4) Break vs. Continue
5) Nested Loops
6) Functions

For Loops

In general, a loop allows a sequence of instructions to execute repeatedly until some condition is met.

Python's *for* loop iterates over items of a sequence(e.g. a list, string or tuple) and process them with some code.

```
for x in sequence:
     block
```

```
In[2]:  for x in [2,3,5,7]:
            print(x, end=“ ”)     # print all on same line
```

```
2 3 5 7
```

range(stop)

A simple use of a *for* loop runs some code a specified number of times using the *range()* function.

range(stop): returns sequence of numbers from 0 (default) up to but not including stop. Increment by 1 (default).

```
In[3]: for i in range(10):
           print(i, end=' ')
```

0 1 2 3 4 5 6 7 8 9

range(start, stop)

range(start, stop): from start up to but not including stop. Increment by 1 (default).

```
In[3]: for i in range(2, 8):
           print(i, end=' ‘)
```

2 3 4 5 6 7

range(start, stop, step)

range(start, stop, step): from start up to but not including stop, increment by step.

```
In[3]: for i in range(1, 10, 2):
           print(i, end=' ')
1 3 5 7 9
```

```
In[3]: for i in range(10, 2, -1):
           print(i, end=' ')
10 9 8 7 6 5 4 3
```

While Loops

A *while* loop executes a block of code while some condition is met.

```
while condition:
    block
```

```
In[6]: i = 0
       while i < 10:
           print(i, end=' ')
           i += 1
0 1 2 3 4 5 6 7 8 9
```

continue vs. break

The *continue* statement is used to skip the current iteration and move to the next iteration whereas the break statement is used to exit a for loop or a while loop.

```
In [7]:  for n in range(20):
             if n % 2 == 0:
                 continue
             print(n, end=' ')
```

```
1 3 5 7 9 11 13 15 17 19
```

continue vs. break

```
In[8]: a,b = 0,1
       amax = 100
       L=[]
       while True:
          a = b
          b = a + b
          if a > amax:
               break
          L.append(a)
       print(L)
```

Create a list with Fibonacci numbers up to amax.

```
[1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]
```

Nested Loops

A *nest loop* is a loop inside of another loop.

```
In[6]: for i in range(1, 4):
           for j in range(1, 5):
               print(i * j, end=‘ ’)
```

```
1 2 3 4
2 4 6 8
3 6 9 12
```

References

1) Vanderplas, Jake, A Whirlwind Tour of Python, O'reilly Media.