Loops – while and for

Next to if decision making tool, the most important programming tool is that of controlled looping. As in life, programming is full of repetition. Programming, often used to model life processes, requires looping controls. An example would be to respectively ask for user input until time to quit. For such control, Python provides while and for looping statements.

General Loop Structure

Short view:

Check weather loop is to start/continue:

Do next loop step

Detailed view:

Check weather loop is to start/continue:

Do one iteration(step) of loop consisting of one or more python statements indented to the same level.

Some statements may be loop (while, for) statements, starting nested loops themselves.

Some statements might be early loop termination such as break or to continue to next iteration such as continue.

The end of indentation is an implied return to this loop's continue check

while loop

The while loop continues while a certain condition holds true. The while continuation test follows the same format as does the if statement.

while statement syntax:

while *test*:

indented loop body

where:

test is a Boolean (true or false) value or an expression, such as x > 2, that evaluates to True or False

indented loop body is one or more Python statements indented to the same indentation, which get executed, in order, once per loop iteration (step).

while examples:

#while\_1.py

var = 1

limit = 5

print("BEFORE: var:", var, "limit:", limit)

while var < limit:

print(" BODY: var:", var, "limit:", limit)

var = var + 1

print("AFTER: var:", var, "limit:", limit)

OUT:

BEFORE: var: 1 limit: 5

BODY: var: 1 limit: 5

BODY: var: 2 limit: 5

BODY: var: 3 limit: 5

BODY: var: 4 limit: 5

AFTER: var: 5 limit: 5

Nested loop within loop

#while\_nested\_simple\_1.py

var = 1

limit = 5

while var < limit:

var2 = 2

limit2 = 4

print("var:", var)

while var2 < limit2:

print("var:", var, "var2:", var2)

var2 = var2 + 1

var = var + 1

OUT:

var: 1

var: 1 var2: 2

var: 1 var2: 3

var: 2

var: 2 var2: 2

var: 2 var2: 3

var: 3

var: 3 var2: 2

var: 3 var2: 3

var: 4

var: 4 var2: 2

var: 4 var2: 3

for loop

for statement Syntax

for for\_variable in for\_list **:**

*loop\_body*

Where:

*for\_variable* is a Python variable which, if not already defined, is created to receive consecutive list values from *for\_list.*

for\_list is a list or an expression such as a function (e.g., range) which returns a list.

loop\_body is one or more Python statements, all indented to the same indentations, which get executed in order, once per loop.

for examples

#for\_1.py

for i in [2, 4, 6, 8]:

print("i:", i)

OUT:

i: 2

i: 4

i: 6

i: 8

#for\_2.py

for i in [2, 4, 6, 8]:

print("i:", i)

print("i\*\*2:", i\*\*2)

OUT:

i: 2

i\*\*2: 4

i: 4

i\*\*2: 16

i: 6

i\*\*2: 36

i: 8

i\*\*2: 64

#for\_range\_0\_limit.py

limit = 8 # Up to but NOT including limit

print("limit:", limit)

for i in range(limit):

print("i:", i)

OUT:

limit: 8

i: 0

i: 1

i: 2

i: 3

i: 4

i: 5

i: 6

i: 7

#for\_range\_start\_limit.py

start = 1

limit = 8

print("start:", start, "limit:", limit)

for i in range(start, limit):

print("i:", i)

OUT:

i: 1

i: 2

i: 3

i: 4

i: 5

i: 6

i: 7

#for\_range\_start\_limit\_inc.py

start = 1 # Starting number

limit = 8 # limit - stop at less

inc = 2 # Increment amount

print("start:", start, "limit:", limit, "inc:", inc)

for i in range(start, limit, inc):

print("i:", i)

OUT:

start: 1 limit: 8 inc: 2

i: 1

i: 3

i: 5

i: 7