Iteration

Loop constructs allow a program to repeat executing sequences of instructions under certain conditions.

The most general form of loop construct is the *while* statement:

Iteration

The most general form of loop constructs is the *while* statement:

```
while <condition>:
  <statements>
```


Iteration

Loop constructs allow a program to repeat executing sequences of instructions under certain conditions.

Iteration

Loop constructs allow a program to repeat executing sequences of instructions under certain conditions.

The most general form of loop construct is the *while* statement:

```
while <condition>:
  <statements>
```

Iteration

The most basic form of loop constructs is the *while* statement:

```
while <condition>:
  <statements>
```

- `<condition>` is a *boolean* expression.
- `<statements>` can be any sequence of statements.

Iteration

The most basic form of loop constructs is the *while* statement:

```
while <condition>:
  <statements>
```

- `<condition>` is a *boolean* expression.

`if` vs. `while`

In the *while* statement

```
while <condition>:
  <statements>
```

`<statements>` is repeatedly executed *over and over again* as long as `<condition>` is `True`.

`if` vs. `while`

In the *if* statement

```
if <condition>:
  <statements>
```

`<statements>` is executed *just once* in case `<condition>` is `True`.

if vs. while

In the *while* statement

```
while <condition>:
  <statements>
```

just like the *if* statement, if `<condition>` is `False` to start with, then `<statements>` is never executed.

if vs. while

In the *while* statement

```
while <condition>:
  <statements>
```

`<statements>` is repeatedly executed *over and over again* as long as `<condition>` is `True`. The loop terminates only when `<condition>` becomes `False`.

Iteration

Example.

```
i = 10
while i > 0:
  print i,
  i = i - 1
```

This simply means that, as long as `i > 0`, print the value of `i` and decrement `i` by `1`.

Meaningful loops

```
while <condition>:
  <statements>
```

In order for this loop to have any value, the value of `<condition>` must become `False` at some point during execution.

Iteration

Example.

```
i = 10
while i > 0:
  print i,
```


Iteration

Example.

```
i = 10
while i > 0:
  print i,
  i = i - 1
```

Output: `10 9 8 7 6 5 4 3 2 1`

Iteration

Example.

```
i = 10
while i > 0:
  print i,
```

Output: `10 10 10 10 10 10 10 10 10...`

Iteration

Example.

```
i = 10
while i > 0:
  print i,
  i = i - 1
```

In order for this to terminate, we need a statement that changes the value of `i > 0` from `True` to `False`.

Iteration

Example.

```
sum = 0
n = input('Enter a number or 0 to stop: ')
while n > 0:
  sum = sum + n
  n = input('Enter a number or 0 to stop: ')
print 'The sum of is', sum
```

The user gets to decide when to terminate by entering a value that makes `n > 0` from `True` to `False`.

Iteration

Example.

```
sum = 0
n = input('Enter a number: ')
while n > 0:
  sum = sum + n
  n = input('Enter a number: ')
print 'The sum of is', sum
```

In this example, there is no statement that explicitly changes the value of `n > 0` from `True` to `False`.

In-class Exercises

- **Write a loop that prints all even numbers.**
- **Write a loop that lets the user input a sequence of numbers, such as 5, 6, 7. The loop should terminate when the next number is out-of-sequence.**
- **Write a method *sum(n)* that returns the sum of 1..n. Use a while loop.**