Conditionals

Conditional constructs allow a program to branch between different sequences of instructions for different cases.

The most basic form of conditional constructs is the *if* statement:

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


Conditionals

The most basic form of conditional constructs is the *if* statement:

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

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

Relational Operators

To compare two numbers, Python uses:

Mathematics	Python
=	==
≠	!=
<	<
>	>
≤	<=
≥	>=

Logical Operators

Three logical operators: `and, or, not`

`x > 0 and y < 10`
(True iff x > 0 and y < 10)

`x > 0 or y < 10`
(True iff either x > 0 or y < 10)

`not(x > 0)`
(True iff x > 0 is false)

Boolean expressions

A *boolean expression* is a code fragment that has a `True/False` value (i.e., its value is either `True` or `False`).

Example.

`0 < 1` (whose value is `True`).

Boolean expressions

A *boolean expression* is a code fragment that has a `True/False` value (i.e., its value is either `True` or `False`).

Example.

`0 < 1`

Boolean expressions

A *boolean expression* is a code fragment that has a `True/False` value (i.e., its value is either `True` or `False`).

Example.

`0 > 1` (whose value is `False`).

Boolean expressions

A *boolean expression* is a code fragment that has a `True/False` value (i.e., its value is either `True` or `False`).

Example.

`0 > 1`

Boolean expressions

A *boolean expression* is a code fragment that has a `True/False` value (i.e., its value is either `True` or `False`).

Example.

`temp >= 60 and temp <= 75`

Boolean expressions

A *boolean expression* is a code fragment that has a `True/False` value (i.e., its value is either `True` or `False`).

Example.

`temp < 32` (assuming `temp` has a value).

Boolean expressions

A *boolean expression* is a code fragment that has a `True/False` value (i.e., its value is either `True` or `False`).

Example.

`temp < 60 or temp > 75`

Boolean expressions

A *boolean expression* is a code fragment that has a `True/False` value (i.e., its value is either `True` or `False`).

Example.

`not (temp < 32)`

Conditionals

The most basic form of conditional constructs is the *if* statement:

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

This simply means that, if `<condition>` is true, execute `<statements>` (otherwise do nothing).

Conditionals

Another important form of conditional constructs is the *if-else* statement:

```
if <condition>:
  <statements1>
else:
  <statements2>
```

This means that, if `<condition>` is true, execute `<statements1>`, or else execute `<statements2>`.

Conditionals

Example.

```
if temp < 32:
  print "It's freezing outside."
```

This simply means that, if `temp < 32`, print `"It's freezing outside."`

Conditionals

Example.

```
if temp < 32:
  print "It's freezing outside."
else:
  print "It's comfortable outside."
```

This means that, if `temp < 32`, print `"It's freezing outside."` Otherwise, print `"It's comfortable outside."`

Conditionals

Nested if-else statements can be simplified with `elif` (which is just an abbreviation of `else if`):

```
if grade >= 90:
  print 'A'
elif grade >= 80:
    print 'B'
elif grade >= 70:
    ...
```


Conditionals

If-else statements can be nested:

```
if grade >= 90:
  print 'A'
else:
  if grade >= 80:
    print 'B'
  else:
    if grade >= 70:
    ...
```


Dangling else Problem

What would this output?

```
x = 7
if x < 5:
  print 'A'
  if x < 3:
    print 'B'
else:
  print 'C'
```

Output: C

Dangling else

What would this output?

```
x = 7
if x < 5:
  print 'A'
  if x < 3:
    print 'B'
else:
  print 'C'
```

Dangling else

What would this output?

```
x = 7
if x < 5:
  print 'A'
  if x < 3:
    print 'B'
  else:
    print 'C'
```

Output: nothing

Dangling else

What would this output?

```
x = 7
if x < 5:
  print 'A'
  if x < 3:
    print 'B'
  else:
    print 'C'
```


Aligning statements

In the *if-else* statement

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

Every line in `<statements>` must be aligned together under the same indentation.

Matching `else` with `if`

In the *if-else* statement

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

`else` is always matched against `if` with the same alignment.

Aligning statements

What would this output?

```
x = 3
if x < 5:
  print 'A',
  print 'B',
else:
  print 'C',
  print 'D',
```

Output: `A, B`

Aligning statements

What would this output?

```
x = 3
if x < 5:
  print 'A',
  print 'B',
else:
  print 'C',
print 'D',
```

Output: `A, B, D`

Aligning statements

What would this output?

```
x = 3
if x < 5:
  print 'A',
  print 'B',
else:
  print 'C',
  print 'D',
```


Aligning statements

What would this output?

```
x = 3
if x < 5:
  print 'A',
  print 'B',
else:
  print 'C',
print 'D',
```

Proper indentation

Compared to other languages (such as C/C++, Java), proper spacing and indentation are very important in Python.

Note. To make indentation, always use the space key, not the tab key.

Long statement

If a single statement does not fit in a single line, do not just move to the next line.

```
print 'The current value is', val,
  'dollars.'
```

This will not work!

Long statement

If a single statement does not fit in a single line, do not just move to the next line.

```
print 'The current value is', val,\
  'dollars.'
```

The end-of-line character \ is required before starting a new line.

Multiple statements

If you need to put multiple statements in a single line, insert ; after each statement.

```
p = p * (1 + apr); print p
```

Exercise

Triangle Rule: If any of the three lengths is greater than the sum of the other two, then you cannot form a triangle. Otherwise, you can.

Write a function named ***is_triangle*** that takes three integers as arguments, and that prints either "Yes" or "No," depending on whether you can or cannot form a triangle from sticks with the given lengths.

Exercise

Definition of an even number: A number is even if it leaves a 0 remainder when divided by 2.

Write a function named ***is_even*** that writes "Yes" if its integer argument is even and otherwise "No"