JÖNKÖPING UNIVERSITY

School of Engineering

CONDITIONAL STATEMENTS IN PYTHON

Peter Larsson-Green

Jönköping University

Autumn 2018

CONDITIONAL STATEMENTS

```
Enter a number: 12
That is a positive number!
```

```
Enter a number: -12
That is a negative number!
```

BOOLEANS

- Used to represent something's correctness.
- Possible values: `True` and `False`.

Examples

`True` `True` → `True`

`False` `False` → `False`

RELATIONAL EXPRESSIONS

Syntax: `<expr1> <operator> <expr2>`

How it is computed

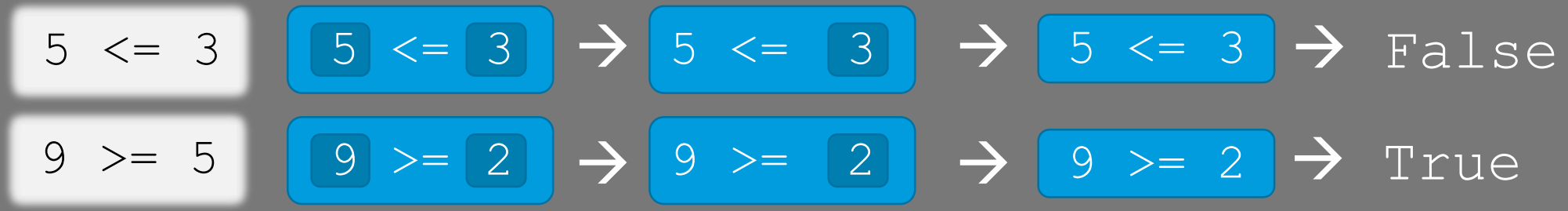
1. Evaluate `<expr1>`.
2. Evaluate `<expr2>`.
3. Apply `<operator>` on the computed values.

Examples

`3 < 5`	`3 < 5` →	`3 < 5` →	`3 < 5` →	`True`
`3 > 5`	`3 > 5` →	`3 > 5` →	`3 > 5` →	`False`
`3 == 2`	`3 == 2` →	`3 == 2` →	`3 == 2` →	`False`
`3 != 2`	`3 != 2` →	`3 != 2` →	`3 != 2` →	`True`

RELATIONAL EXPRESSIONS

Examples



THE IF STATEMENT

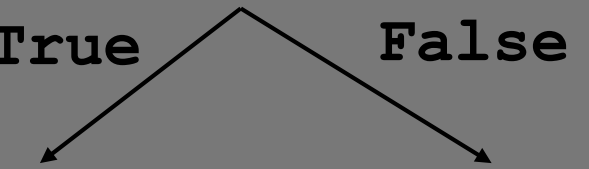
Conditionally executes statements.

Syntax

```
if <expr> :
    Statement 1
    Statement 2
    Statement ...
```

How it is executed

1. Evaluate `<expr>`.



2. Execute `<statementX>`.
3. Go to next statement.

2. Go to next statement.

THE ELIF STATEMENT

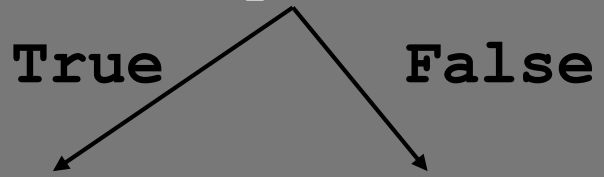
- `elif` is short for else if.
- Optional continuation of an if/elif statement.

Syntax

```
if <expr-a> :
    Statements-a
elif <expr-b> :
    Statements-b
```

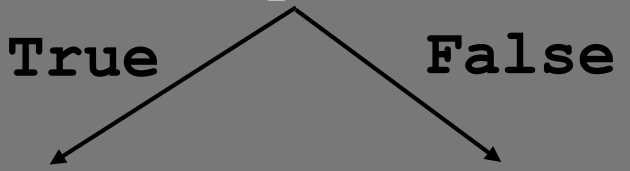
How it is executed

1. Evaluate `<expr-a>`.



2. Execute `<statements-a>`.
3. Go to next statement.

2. Evaluate `<expr-b>`.



3. Execute `<statements-b>`.
4. Go to next statement.

3. Go to next statement.

THE ELIF STATEMENT

```
if <expr-a> :
    Statements-a
elif <expr-b> :
    Statements-b
```

```
if <expr-a> :
    Statements-a
elif <expr-b> :
    Statements-b
elif <expr-c> :
    Statements-c
```

```
if <expr-a> :
    Statements-a
elif <expr-b> :
    Statements-b
elif <expr-c> :
    Statements-c
elif <expr-d> :
    Statements-d
```

THE ELSE STATEMENT

Optional tail to an if/elif statement.

Syntax

```
if <expr-a> :
    Statements-a
else:
    Statements-b
```

```
if <expr-a> :
    Statements-a
elif <expr-b> :
    Statements-b
else:
    Statements-c
```

How it is executed

If all `<expr-X>` evaluates to `False`, execute the else statements.

EXAMPLE

```
def is_between_5_8(x):
  if x < 5:
    return False
  elif 8 < x:
    return False
  else:
    return True
```

```
def is_between_5_8(x):
  if 5 <= x:
    if x <= 8:
      return True
    else:
      return False
  else:
    return False
```

```
is_between_5_8(4) → False
is_between_5_8(5) → True
```

```
is_between_5_8(8) → True
is_between_5_8(9) → False
```

EXAMPLE

```
def is_between_5_8(x):
  if x < 5:
    return False
  if 8 < x:
    return False
  return True
```

```
def is_between_5_8(x):
  if 5 <= x:
    if x <= 8:
      return True
  return False
```

```
is_between_5_8(4) → False
is_between_5_8(5) → True
```

```
is_between_5_8(8) → True
is_between_5_8(9) → False
```

EXAMPLE

```
def max(number_a, number_b):
  if number_a < number_b:
    return number_b
  else:
    return number_a

four = max(3, 4)
nine = max(9, 6)
```

```
def max(number_a, number_b):
  if number_a < number_b:
    return number_b
  return number_a
```

```
def max(number_a, number_b):
  biggest = number_a
  if number_a < number_b:
    biggest = number_b
  return biggest
```

THE IF-ELSE EXPRESSION

Syntax

```
<expr1> if <expr2> else <expr3>
```

How it is evaluated

1. Evaluate `<expr2>`.

2. Evaluate `<expr1>` and yield the result.

2. Evaluate `<expr3>` and yield the result.

THE IF-ELSE EXPRESSION

```
variable = <expr2> if <expr1> else <expr3>
```

```
if <expr1>:
  variable = <expr2>
else:
  variable = <expr3>
```

```
def func():
  if <expr1>:
    return <expr2>
  else:
    return <expr3>
```

```
def func():
  return <expr2> if <expr1> else <expr3>
```

EXAMPLE

```
def max(number_a, number_b):
  return number_b if number_a < number_b else number_a
```

THE NOT EXPRESSION

Inverts boolean values.

Syntax: `not <expr>`

How it is computed

1. Evaluate `<expr>`.
2. Invert that value.

Examples

`not True` → `not True` → `False`

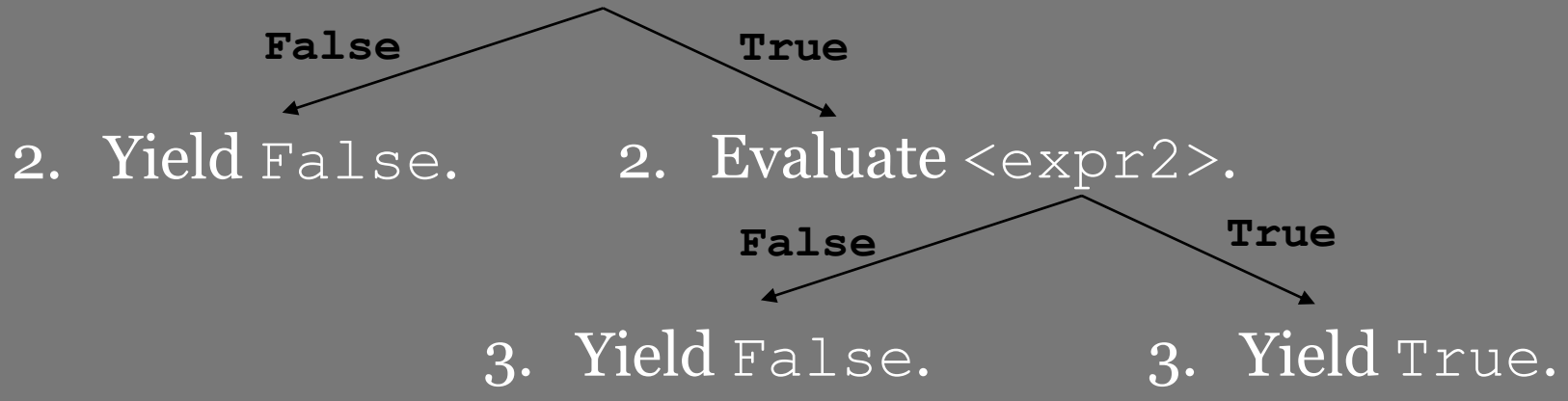
`not False` → `not False` → `True`

THE AND EXPRESSION

Syntax: `<expr1> and <expr2>`

How it is computed

1. Evaluate `<expr1>`.



THE AND EXPRESSION

Syntax: `<expr1> and <expr2>`

`False and False` → `False and False` → `False`

`False and True` → `False and True` → `False`

`True and False` → `True and False` → `True and False` → `False`

`True and True` → `True and True` → `True and True` → `True`

EXAMPLE

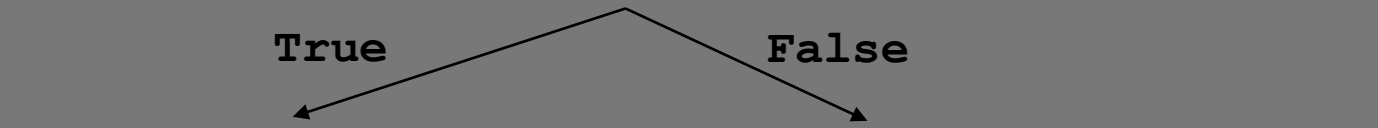
```
def is_between_2_5(x):
  return 2 < x and x < 5


yes = is_between_2_5(4.95)
no = is_between_2_5(0)
```

THE OR EXPRESSION

Syntax: `<expr1>` `or` `<expr2>`

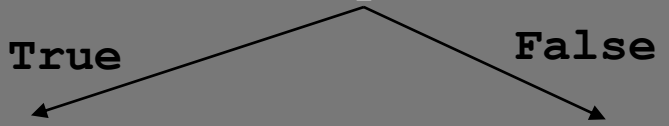
How it is computed

1. Evaluate `<expr1>`.



2. Yield `True`.

2. Evaluate `<expr2>`.



3. Yield `True`.

3. Yield `False`.

THE OR EXPRESSION

Syntax: `<expr1> or <expr2>`

`False or False` → `False or False` → `False or False` →

`False`

`False or True` → `False or True` → `False or True` →

`True`

`True or False` → `True or False` → `True`

`True or True` → `True or True` → `True`

EXAMPLE

```
def is_not_between_2_5(x):
  return x < 2 or 5 < x


yes = is_not_between_2_5(0)
no = is_not_between_2_5(4.95)
```