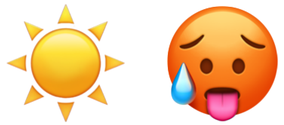
Practical Python for Beginners

CONDITIONALS

Sarah Holderness
PLURALSIGHT AUTHOR
@dr_holderness

How Do We Make Decisions in a Program?

A conditional statement, or if statement, lets us make decisions in Python



If it's sunny and 90° or higher

Stay inside!

If it's raining

Stay inside!

Otherwise ...

Enjoy the outdoors!

To make decisions we need to make comparisons inside if statements.

The 6 Python Comparators

<	<=	==	>=	>	!=
less than	less than equal to	equal	greater than equal to	greater than	NOT equal

Making a comparison is like asking the question: Is the temperature equal to 95?

```
>>> temp = 95
>>> temp == 95
True
```

Assigning 95 to the temp variable.

Notice that assignment is 1 = sign. And the equals to comparator is 2 == signs.

The 6 Python Comparators

<	<=	==	>=	>	!=
less than	less than equal to	equal	greater than equal to	greater than	NOT equal

```
>>> temp = 95
>>> temp == 95
True
```

Is the temperature less than 90?

```
>>> temp < 90
False
```

An `if` Statement

An if statement lets us **decide** what to do: if **True**, then ***do*** this.

weather.py

```
temperature = 95

if temperature > 80:

```

Assign `95` *to the* `temperature` *variable*

Is temperature greater than `80`*?*

If this is `True`*, let's add something to do here*

An `if` Statement

An if statement lets us **decide** what to do: if **True**, then ***do*** this.

weather.py

```
temperature = 95

if temperature > 80:
    print("It's too hot!")
    print("Stay inside!")
```

True

So these lines are run

```
> python3 weather.py
It's too hot!
Stay inside!
```

`if` Code Block

weather.py

```
temperature = 95

if temperature > 80:
    print("It's too hot!")
    print("Stay inside!")
```

Any indented code that comes after an if statement is called a code block

```
> python3 weather.py
It's too hot!
Stay inside!
```

When the `if` Statement is `False`

weather.py

```
temperature = 75

if temperature > 80:
    print("It's too hot!")
    print("Stay inside!")
```

False

So these lines are NOT run

```
> python3 weather.py
```

And there is no output

The Program Continues After the `if` Code Block

weather.py

```
temperature = 75

if temperature > 80:
    print("It's too hot!")
    print("Stay inside!")
print("Have a good day!")
```

False

```
> python3 weather.py
Have a good day!
```

The program keeps running after the if statement and its code block, so this is printed after.

Rules for Whitespace in Python

weather.py

```
temperature = 75
if temperature > 80:
  print("It's too hot!")
    print("Stay inside!")
```

2 spaces indent

4 spaces indent

```
$ python3 weather.py
  File "weather.py", line 6
    print("Stay inside!")
    ^
IndentationError: unexpected indent
```

Whitespace indents in Python need to be consistent, otherwise there will be an IndentationError.

An `if`, `else` Statement

weather.py

```
temperature = 75

if temperature > 80:
    print("It's too hot!")
    print("Stay inside!")
```

False

How do we do something else here if this is False?

An `if`, `else` Statement

weather.py

```
temperature = 75

if temperature > 80:
    print("It's too hot!")
    print("Stay inside!")
else:
    print("Enjoy the outdoors!")
```

If this statement is True, then run the code block below

Otherwise, then run the code block below

```
> python3 weather.py
Enjoy the outdoors!
```

if, elif, and else

weather.py

```
temperature = 50

if temperature > 80:
    print("It's too hot!")
    print("Stay inside!")
elif temperature < 60:
    print("It's too cold!")
    print("Stay inside!")
else:
    print("Enjoy the outdoors!")
```

False

True

So both of these lines are run.

```
> python3 weather.py
It's too cold!
Stay inside!
```

Can We Combine Two `if` Statements?

weather.py

```
temperature = 75

if temperature > 80:
    print("Stay inside!")
elif temperature < 60:
    print("Stay inside!")
else:
    print("Enjoy the outdoors!")
```

What if we wanted to shorten our program to only say: "Stay inside!" OR "Enjoy the outdoors!"

We're repeating the `print("Stay inside!")` *twice*

Can we combine the first 2 `if` *statements?*

Logical Operator - **or**

weather.py

```
temperature = 75

if temperature > 80 or temperature < 60:
    print("Stay inside!")
else:
    print("Enjoy the outdoors!")
```

The keyword `or` *lets you combine multiple comparisons. At least* ***one*** *needs to be* `True` *for the whole if statement to be* `True`

Logical Operator - **or**

Only **one** comparison needs to be True for the if statement to be True

weather.py

```
temperature = 75

if temperature > 80 or temperature < 60:
    print("Stay inside!")
else:
    print("Enjoy the outdoors!")
```

False or False → False

This is run

```
> python3 weather.py
Enjoy the outdoors!
```

Logical Operator - **or**

Only **one** comparison needs to be True for the if statement to be True

weather.py

```
temperature = 50

if temperature > 80 or temperature < 60:
    print("Stay inside!")
else:
    print("Enjoy the outdoors!")
```

False *or* *True* → *True*

This is run

```
> python3 weather.py
Stay inside!
```

Store the Forecast as a String

weather.py

```
temperature = 75
forecast = "rain"
```

Let's add another variable with the forecast as "rainy", "cloudy", or "sunny".

Logical Operator - **and**

weather.py

```
temperature = 75
forecast = "rain"

if temperature < 80 and forecast != "rain":
    print("Go outside!")
else:
    print("Stay inside!")
```

The keyword `and` *also lets you combine multiple comparisons. But* ***both*** *need to be* `True` *for the whole if statement to be* `True`

Logical Operator - **and**

Both comparisons need to be True for the if statement to be True

weather.py

```
temperature = 75
forecast = "rain"

if temperature < 80 and forecast != "rain":
    print("Go outside!")
else:
    print("Stay inside!")
```

True *and* *False* → *False*

This is run

Logical Operator - **and**

Both comparisons need to be True for the if statement to be True

weather.py

```
temperature = 75
forecast = "sunny"

if temperature < 80 and forecast != "rain":
    print("Go outside!")
else:
    print("Stay inside!")
```

Logical Operator - **not**

weather.py

```
forecast = "rain"


if not forecast == "rain":
    print("Go outside!")
else:
    print("Stay inside!")
```

The keyword `not` *lets you negate a comparison. And can help make the statement more readable.*

Logical Operator - **not**

The keyword **not** negates a comparison

weather.py

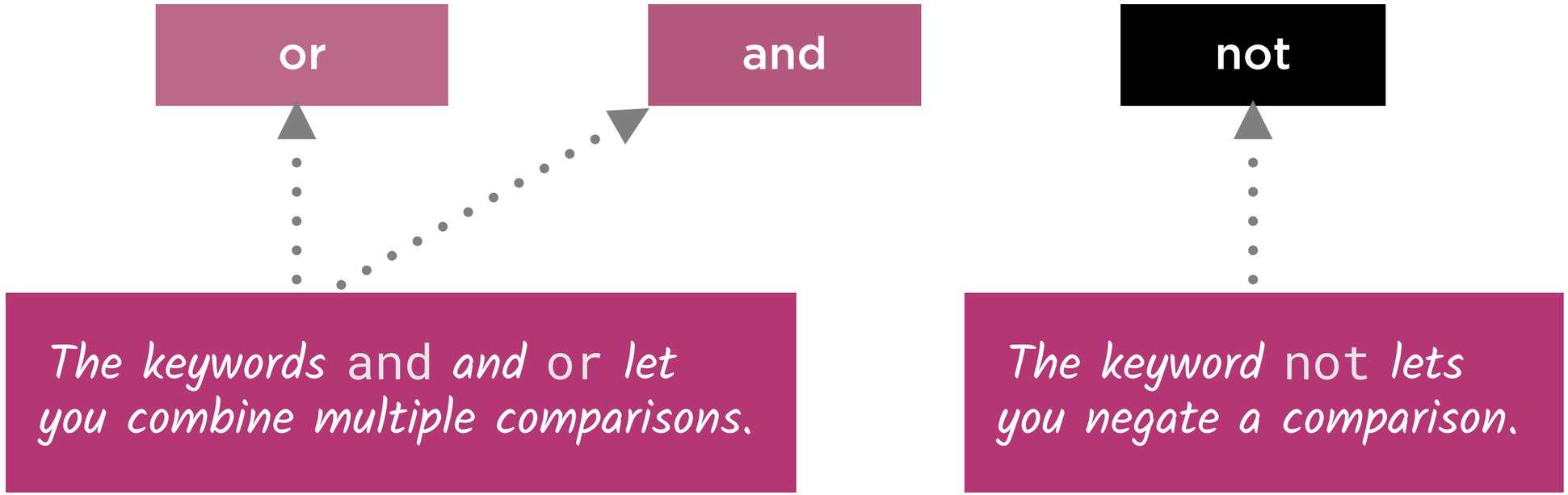
```
forecast = "rain"

if not forecast == "rain":
    print("Go outside!")
else:
    print("Stay inside!")
```

not *True* → *False*

Negate means make the opposite:
not True becomes False,
not False becomes True

This is run

The 3 Python Logical Operators



All of the Primitive Data Types

int

```
>>> amount = 10
```

float

```
>>> tax = .06
```

string

```
>>> name = "Sarah"
```

boolean

```
>>> answer = True
```

A `boolean` *can store a* `True` *or* `False` *value.*

Evaluating **Boolean** Variables

weather.py

```
raining = True

if raining:
    print("Stay inside!")
```

You can set boolean variables to either True or False

This reads like English

```
> python3 weather.py
Stay inside!
```

Evaluating **Boolean** Variables

weather.py

```
raining = True

if not raining:
    print("Go outside!")
else:
    print("Stay inside!")
```

not *True* → *False*

This is run

```
> python3 weather.py
Stay inside!
```

END OF SECTION

Building Your First Application with Python

IMPORTING MODULES

Sarah Holderness
PLURALSIGHT AUTHOR
@dr_holderness

A Random Rock, Paper, Scissors Game

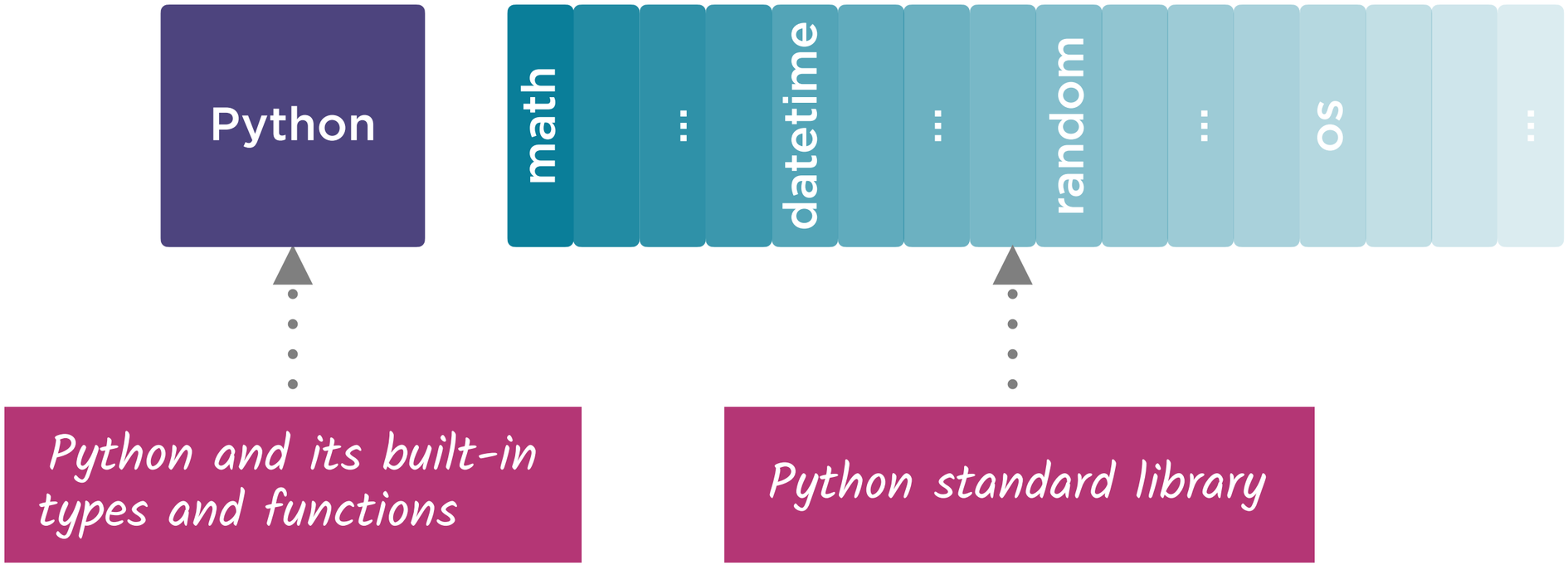
The computer has the same choice every time

How can we randomly pick the computer's choice?

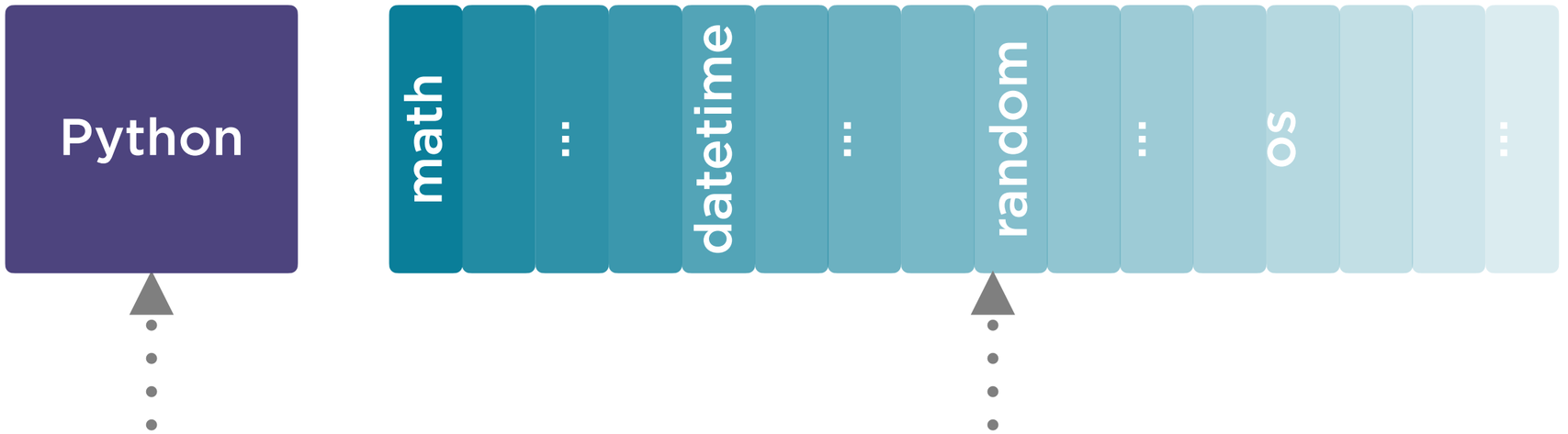
```
computer_choice = 'scissors'

user_choice = input("Do you want - rock, paper, or scissors?\n")
```

The user gets to pick a new choice for each game

When You Install Python



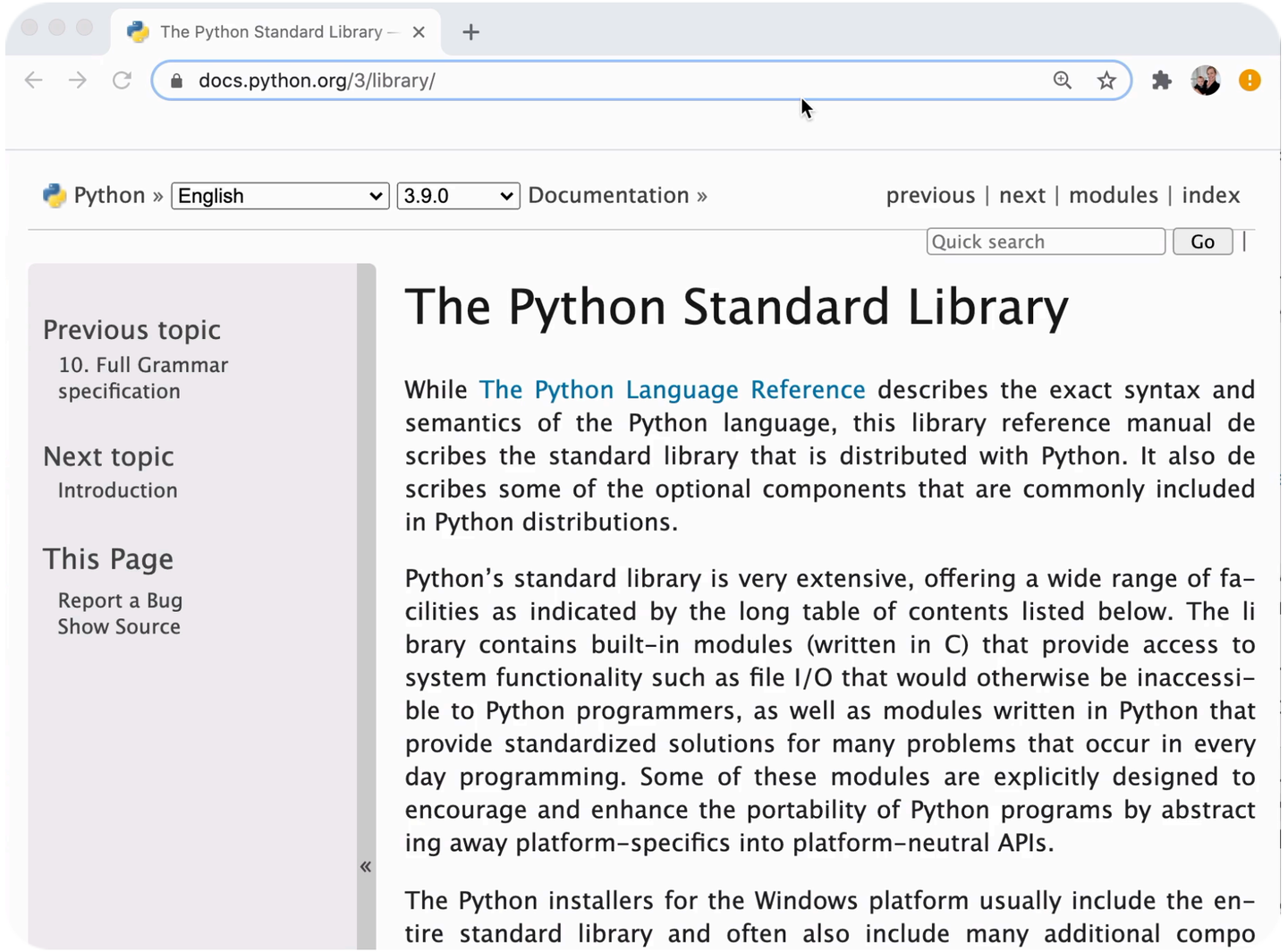
The Standard Library



Our programs so far have just used Python and its built-in types and functions

But if you need something extra you can import it from the Python standard library

Looking Up the `random` Module



Using the Random Module

roll_dice.py

```
import random

roll = random.randint(1,6)
```

We need to import the module to use it

This function will return a random number between 1 and 6

Printing Out the Result

roll_dice.py

```
import random

roll = random.randint(1,6)

print("The computer rolled a " + str(roll))
```

Don't forget to convert the int to a string to concatenate it.

```
> python3 roll_dice.py
The computer rolled a 6
```

If we ran this more times we would see different random numbers generated.

Guess the What the Computer Rolled

roll_dice.py

```
import random

roll = random.randint(1,6)

guess = int(input('Guess the dice roll:\n'))
```

We want to convert the input to an int *so we can compare* guess *to* roll.

```
> python3 roll_dice.py
Guess the dice roll:
6
```

Guess the What the Computer Rolled

roll_dice.py

```
import random

roll = random.randint(1,6)

guess = int(input('Guess the dice roll:\n'))

if guess == roll:
    print("Correct! They rolled a " + str(roll))
```

```
> python3 roll_dice.py
Guess the dice roll:
6
Correct! They rolled a 6
```

Guess the What the Computer Rolled

roll_dice.py

```
import random

roll = random.randint(1,6)

guess = int(input('Guess the dice roll:\n'))

if guess == roll:
    print("Correct! They rolled a " + str(roll))
```

```
> python3 roll_dice.py
Guess the dice roll:
6
```

Why isn't there output now?

We need an else statement for when the guess is wrong.

Guess the What the Computer Rolled

roll_dice.py

```
import random

roll = random.randint(1,6)

guess = int(input('Guess the dice roll:\n'))

if guess == roll:
    print("Correct! They rolled a " + str(roll))
else:
    print("Wrong! They rolled a " + str(roll))
```

```
> python3 roll_dice.py
Guess the dice roll:
6
Wrong! They rolled a 4
```