

Practical Python for Beginners

CONDITIONALS



Sarah Holderness
PLURALSIGHT AUTHOR
[@dr_holderness](https://twitter.com/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 <...>
```

*Assign 95 to the
temperature variable*

```
if temperature > 80: <...>
```

*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: <... True
```

```
    print("It's too hot!")
```

```
    print("Stay inside!")
```

*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!")
```

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

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



When the if Statement is False

weather.py

```
temperature = 75
```

```
if temperature > 80: <... False
```

```
    print("It's too hot!") <... So these lines are NOT run
```

```
    print("Stay inside!")
```

```
> python3 weather.py
```

And there is no output



The Program Continues After the if Code Block

weather.py

```
temperature = 75  
  
if temperature > 80:    ◀ ••• False  
    print("It's too hot!")  
    print("Stay inside!")  
  
    print("Have a good day!") ◀ •••
```

```
> 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!") ◀··· 2 spaces indent
    print("Stay inside!") ◀··· 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:    ◀••• False  
    print("It's too hot!")  
    print("Stay inside!")
```

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:    ◀ ••• False
    print("It's too hot!")
    print("Stay inside!")

elif temperature < 60:  ◀ ••• True
    print("It's too cold!")
    print("Stay inside!")

else:
    print("Enjoy the outdoors!")
```

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

*So both of these
lines are run.*



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
```

False

or

False



False

```
if temperature > 80 or temperature < 60:
```

```
    print("Stay inside!")
```

```
else:
```

```
    print("Enjoy the outdoors!")
```

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
```

False

or

True

→ True

```
if temperature > 80 or temperature < 60:
```

```
    print("Stay inside!")
```

```
else:
```

```
    print("Enjoy the outdoors!")
```



... 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"
```

>

True

and

False



False

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

← · · 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"
```

>

True

and

True



True

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

← ..

This is run

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"
```

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

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

This is run

The 3 Python Logical Operators

or

and

not

The keywords and and or let you combine multiple comparisons.

The keyword not lets you negate a comparison.

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*

> python3 weather.py
Stay inside!



Evaluating Boolean Variables

weather.py

```
raining = True
```

not

True



False

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

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](https://twitter.com/dr_holderness)

A Random Rock, Paper, Scissors Game

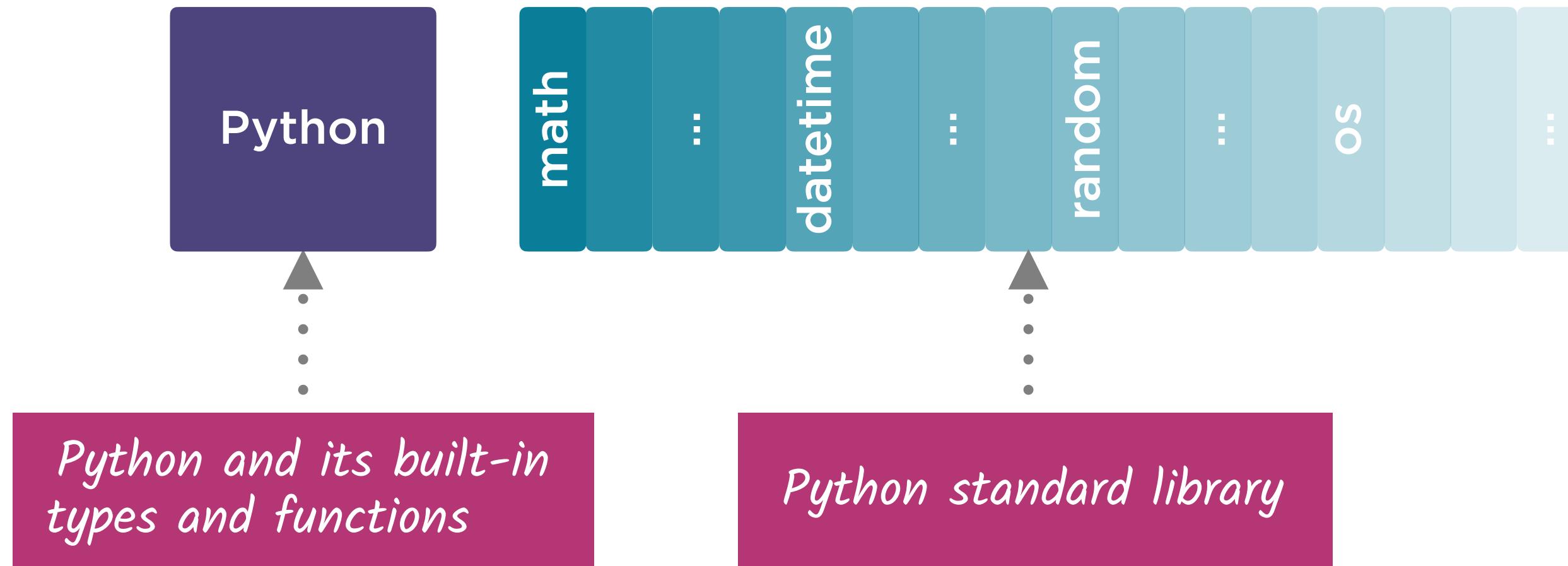
The computer has the same choice every time

How can we randomly pick the computer's choice?

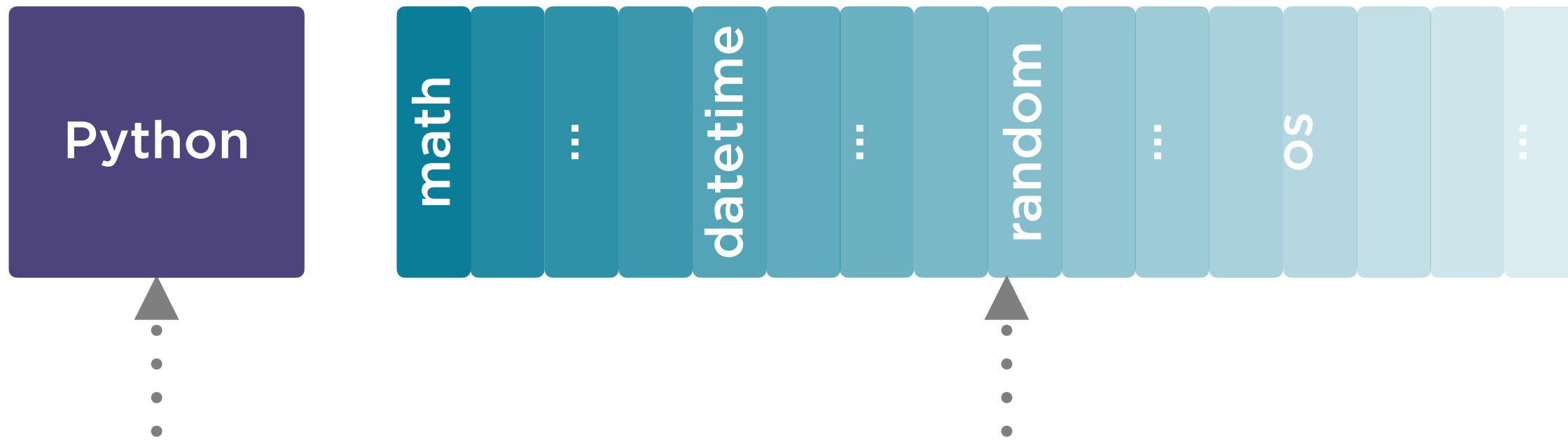
```
computer_choice = 'scissors'  
user_choice = input("Do you want - rock, paper, or scissors?\n")
```

The user gets to pick a new choice for each game

When You Install Python



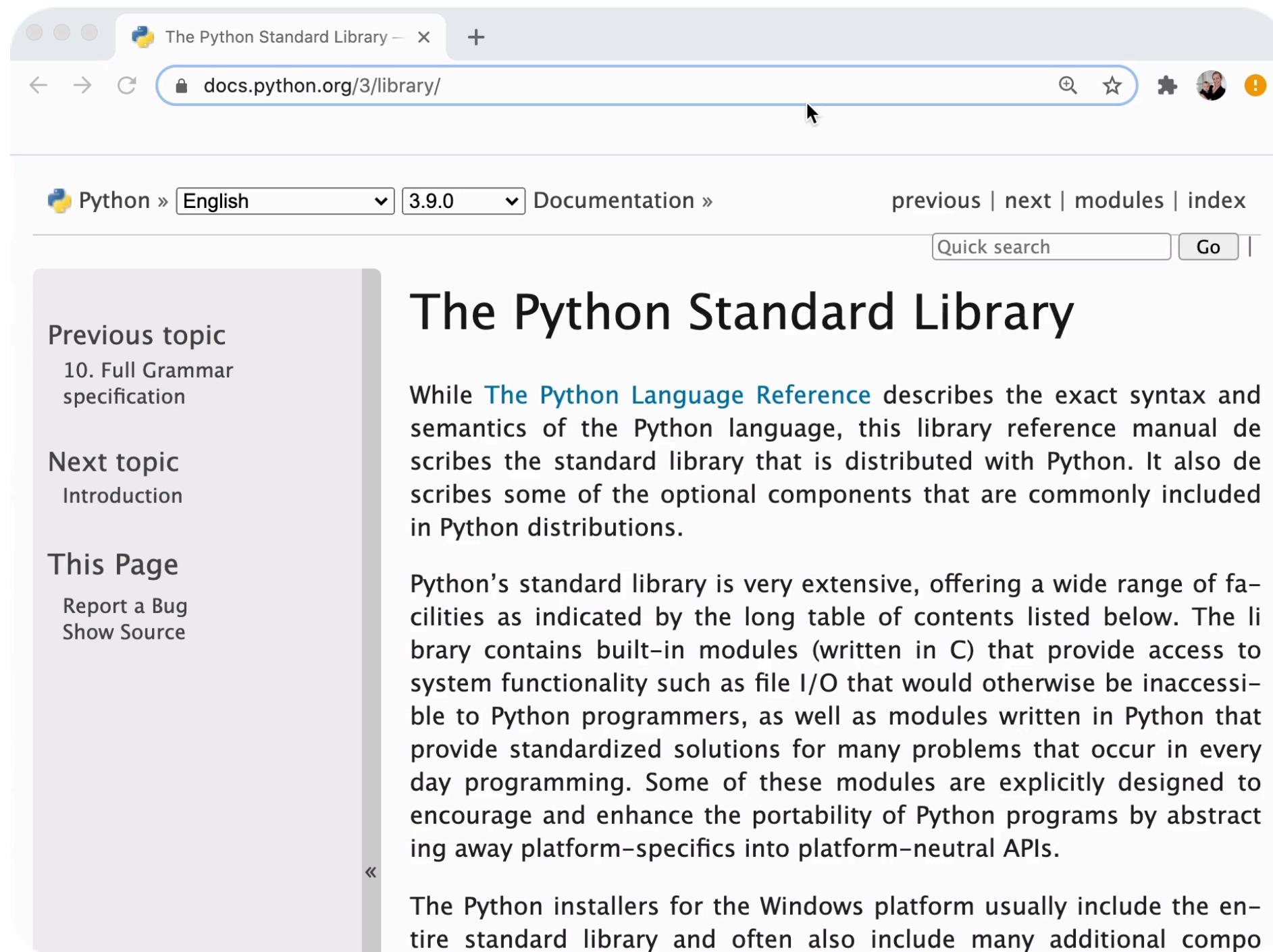
The Standard Library



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



The screenshot shows a web browser window displaying the Python Standard Library documentation. The title bar reads "The Python Standard Library - docs.python.org/3/library/". The main content area is titled "The Python Standard Library". It includes a sidebar with links to "Previous topic" (10. Full Grammar specification), "Next topic" (Introduction), and "This Page" (Report a Bug, Show Source). The main text describes the standard library and its extensive facilities. A footer note mentions Windows installers.

The Python Standard Library

While [The Python Language Reference](#) describes the exact syntax and semantics of the Python language, this library reference manual describes the standard library that is distributed with Python. It also describes some of the optional components that are commonly included in Python distributions.

Python's standard library is very extensive, offering a wide range of facilities as indicated by the long table of contents listed below. The library contains built-in modules (written in C) that provide access to system functionality such as file I/O that would otherwise be inaccessible to Python programmers, as well as modules written in Python that provide standardized solutions for many problems that occur in everyday programming. Some of these modules are explicitly designed to encourage and enhance the portability of Python programs by abstracting away platform-specifics into platform-neutral APIs.

The Python installers for the Windows platform usually include the entire standard library and often also include many additional compo

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
```

