



Welcome to Lesson 5: Functions and Collections

▶ Today's Lesson

- ▶ Collections
 - ▶ Negative Indexing
 - ▶ Sets
 - ▶ Tuples
 - ▶ Dictionaries
- ▶ Functions

▶ Collections

- ▶ “Collection” of items
- ▶ Hold multiple values

▶ Immutable vs Mutable

- ▶ Mutable: single elements can be changed after creation
 - ▶ `listName[index] = newValue`
- ▶ Immutable: single elements cannot be changed after creation

Review List

- ▶ Syntax: [value1, value2, ...]
- ▶ Mutable
- ▶ Allows duplicates
- ▶ Indexing

Index	0	1	2	3	4	5
Value	"hello"	3	True	11.2	21	'hi'

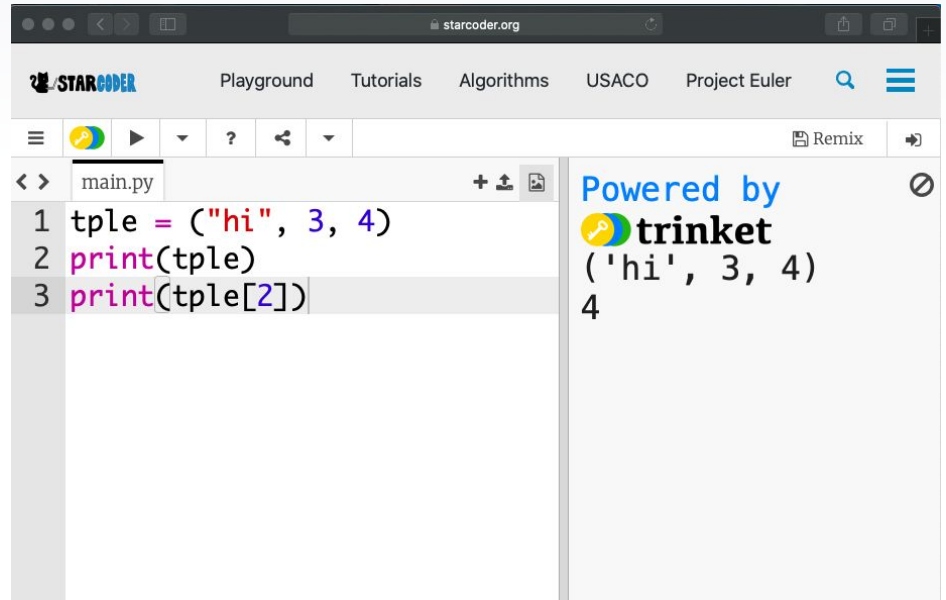
▶ Negative Indexing

- ▶ Finding the value from the end

Index	0 or -6	1 or -5	2 or -4	3 or -3	4 or -2	5 or -1
Value	"hello"	3	True	11.2	21	'hi'

▶ Tuple

- ▶ Syntax: `(value1, value2, ...)`
- ▶ Immutable
- ▶ Allows duplicates
- ▶ Indexing



The screenshot shows a web browser window with the URL `starcoder.org`. The page has a navigation bar with links for "Playground", "Tutorials", "Algorithms", "USACO", and "Project Euler". Below the navigation bar is a toolbar with icons for running, help, and sharing. The main area is a code editor with a file named `main.py`. The code in the editor is:

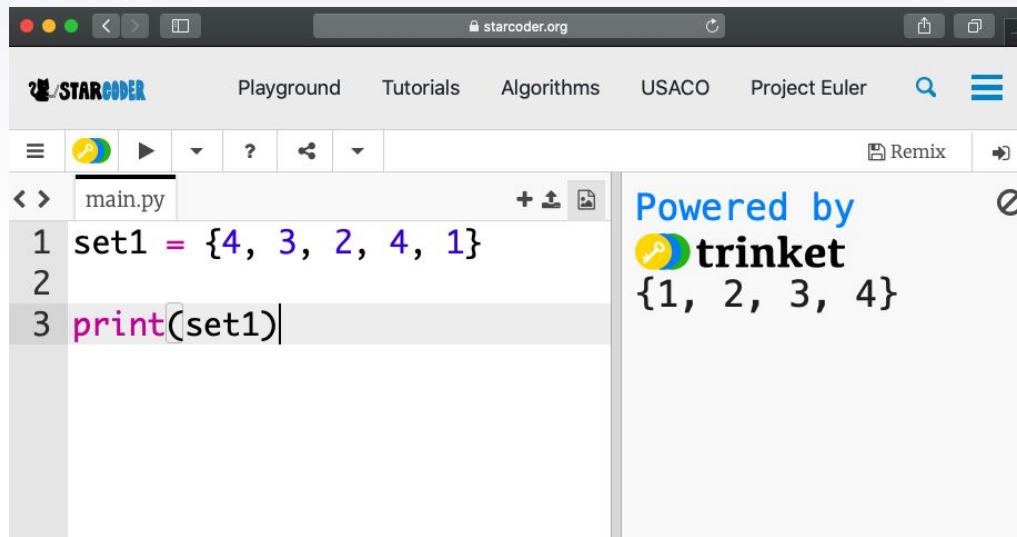
```
1 tple = ("hi", 3, 4)
2 print(tple)
3 print(tple[2])
```

To the right of the code editor is a terminal window showing the output of the code:

```
Powered by trinket
('hi', 3, 4)
4
```

Set

- ▶ Syntax: {value1, value2, ...}
- ▶ Mutable
- ▶ Does not allows duplicates
- ▶ No indexing



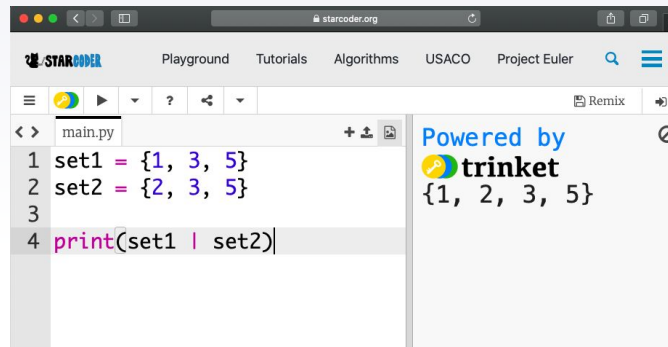
The screenshot shows a web browser window at starcoder.org. The page has a navigation bar with links for Playground, Tutorials, Algorithms, USACO, and Project Euler. Below the navigation bar is a toolbar with icons for running, help, and sharing. The main area is a code editor with a file named 'main.py'. The code contains three lines: '1 set1 = {4, 3, 2, 4, 1}', '2', and '3 print(set1)'. To the right of the code editor is a panel titled 'Powered by trinket' which displays the output of the code: '{1, 2, 3, 4}'.

```
main.py  
1 set1 = {4, 3, 2, 4, 1}  
2  
3 print(set1)
```

Powered by trinket
{1, 2, 3, 4}

Set Operations

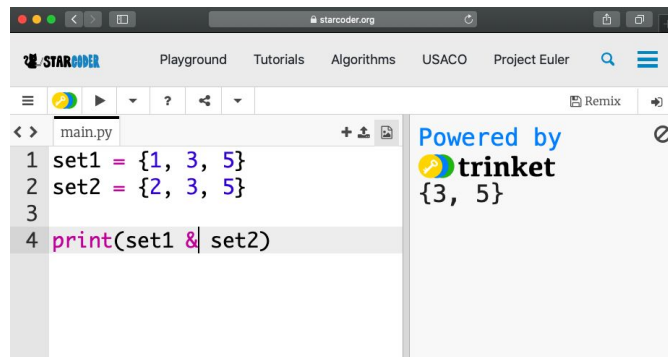
- ▶ `|` or `union()`
 - ▶ Works as set **or**
- ▶ `&` or `intersection()`
 - ▶ Works as set **and**



A screenshot of the StarCoder playground interface. The code editor shows a Python script named 'main.py' with the following code:

```
1 set1 = {1, 3, 5}
2 set2 = {2, 3, 5}
3
4 print(set1 | set2)
```

The output panel on the right displays the result of the union operation: `{1, 2, 3, 5}`. The interface includes a navigation bar with links to 'Playground', 'Tutorials', 'Algorithms', 'USACO', and 'Project Euler', and a 'Remix' button.

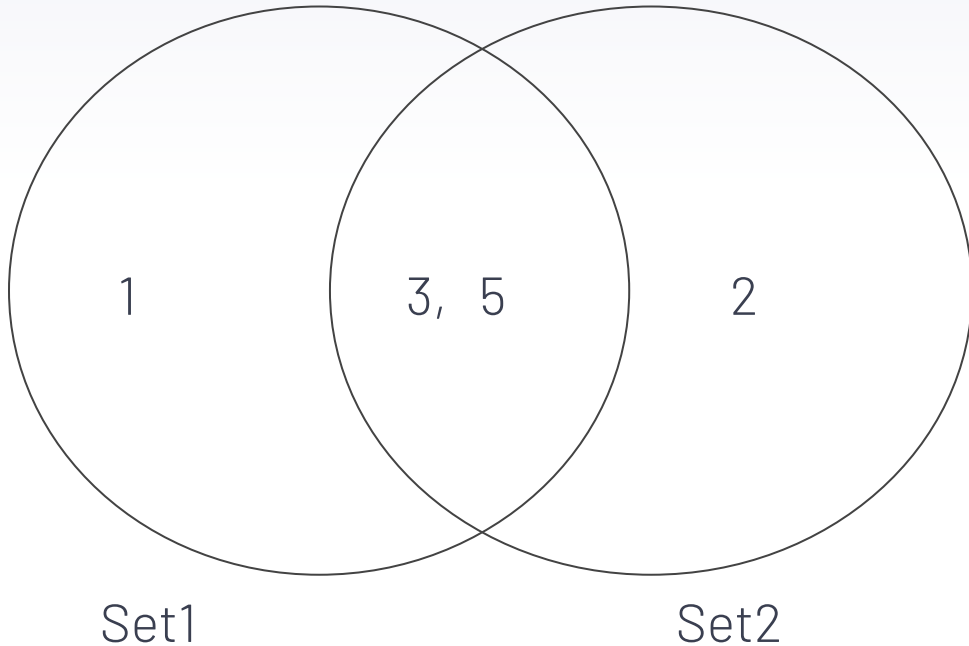


A screenshot of the StarCoder playground interface. The code editor shows a Python script named 'main.py' with the following code:

```
1 set1 = {1, 3, 5}
2 set2 = {2, 3, 5}
3
4 print(set1 & set2)
```

The output panel on the right displays the result of the intersection operation: `{3, 5}`. The interface includes a navigation bar with links to 'Playground', 'Tutorials', 'Algorithms', 'USACO', and 'Project Euler', and a 'Remix' button.

Set Operations



```
main.py
1 set1 = {1, 3, 5}
2 set2 = {2, 3, 5}
3
4 print(set1 | set2)
```

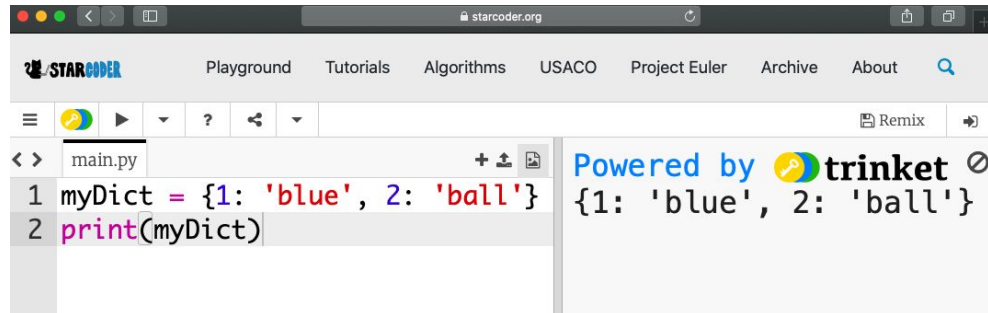
Powered by trinket
{1, 2, 3, 5}

```
main.py
1 set1 = {1, 3, 5}
2 set2 = {2, 3, 5}
3
4 print(set1 & set2)
```

Powered by trinket
{3, 5}

Dictionary

- ▶ Syntax: {key1:value1, key2:value2, ...}
- ▶ Mutable
- ▶ Does not allows duplicates
- ▶ Indexing



The screenshot shows a web browser window with the URL 'starcoder.org'. The page has a navigation bar with links for 'Playground', 'Tutorials', 'Algorithms', 'USACO', 'Project Euler', 'Archive', and 'About'. Below the navigation bar is a toolbar with icons for running, saving, and other actions. The main area is a code editor with a file named 'main.py' containing the following Python code:

```
1 myDict = {1: 'blue', 2: 'ball'}
2 print(myDict)
```

The output of the code is displayed on the right side of the editor, showing the dictionary representation: {1: 'blue', 2: 'ball'}. The output is powered by 'trinket'.

► Collections Review

Data Type	Ordered?	Mutable ?	Indexed ?	Allow Duplicate Member ?
List	Yes	Yes	Yes	Yes
Tuple	Yes	No	Yes	Yes
Set	No	Yes	No	No
Dictionary	No	Yes	Yes	No

▶ Functions

- ▶ Function - group of related statements that executes only when called
 - ▶ Math function: $f(x)$
- ▶ Doesn't execute unless "called"

Function Syntax

Keyword to
define function

Unique name of
function

Parameters - arguments/values
that are passed into function.
Optional

```
def function_name(parameters):  
    statement  
    return value
```

The diagram illustrates the function syntax with three labels and arrows pointing to the corresponding parts of the code: 'Keyword to define function' points to 'def', 'Unique name of function' points to 'function_name', and 'Parameters - arguments/values that are passed into function. Optional' points to 'parameters'.

```
X = function_name(parameters)
```

▶ Function Syntax

```
def function_name(parameters):  
    statement  
    return value
```

Statement(s) executed only if
function ran

←

What function prints/will give

```
X = function_name(parameters)
```

←

Calling the function. Multiple times!

► Function Example

```
def print_animal(animalname):
```

```
    if (animalname == "cat"):
```

```
        return 1
```

```
    else:
```

```
        return 0
```


```
print(print_animal("cat"))
```

```
print(print_animal(2))
```





Examples and Problems!

Type Your Answers in the Chat



```
def hello():  
    greetString = "Hello, " + "World!"  
    print(greetString)
```

```
hello()
```




```
def hello(name):  
    greetString = "Hello, " + name  
    return greetString  
  
print(hello('World!'))
```



```
fruits = ("apple", "banana", "orange")
```

```
print(fruits[-2])
```


What does this print?



```
fruits = {'apple', 'banana', 'orange'}
```


```
for a in fruits:  
    print(a)
```

What type is fruits?



```
fruits = ("apple", "banana", "orange")  
print(fruits)
```

What type is fruits?



```
fruits = {'apple', 'banana', 'orange'}
```

```
a = 'carrot'
```

```
if a in fruits:
```

```
    print(True)
```

```
else:
```

```
    print(False)
```

What does this print?