POWERING POTENTIAL

Lists and Strings

WELCOME

Insert a RECENT and HIGH QUALITY photo of yourself here

Trainer Name

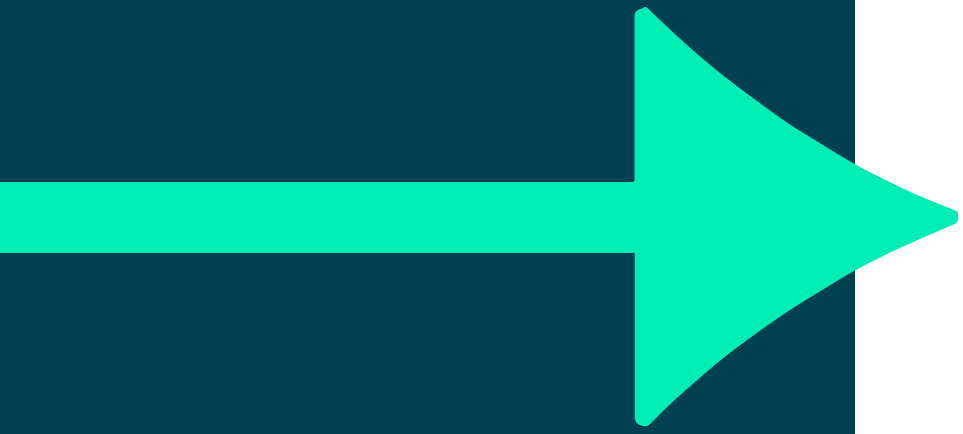
Trainer Role, QA

- A few key facts...
- Previous role
- Qualifications

SESSION OVERVIEW

- Engage in practical activities to support module evidence collection
- Introductions and ice-breaker activity
- Provide an overview of the 3-day class-based learning
- Provide support and guidance for the successful completion of Module 4B

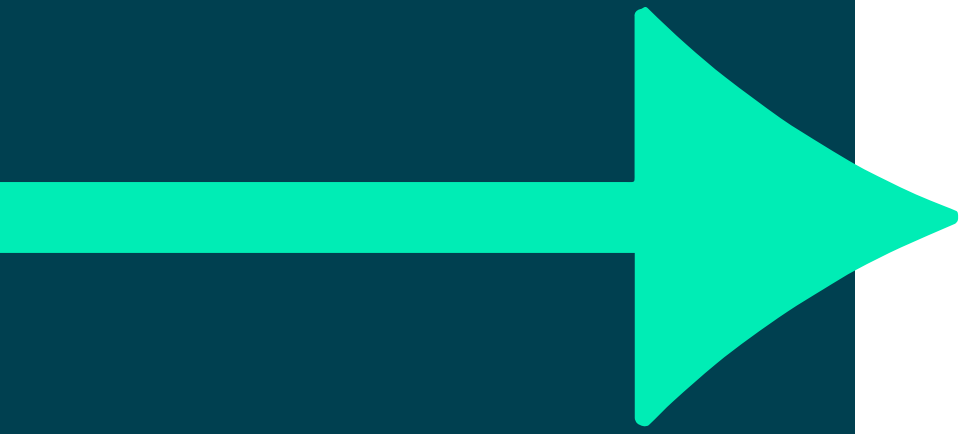
LESSON OBJECTIVES



In this chapter, you'll learn how to:

- Create lists
- Read an item from a list
- Read multiple items from a list (slicing)
- Add an item to a list
- Remove an item from a list
- Replace an item
- Strings as lists of characters
- String methods

PYTHON LISTS



Lists store multiple values (elements)

```
numbers = [1,3,5,7]
```

1
3
5
7

```
names = ['Bob', 'Steve', 'Helen']
```

Bob
Steve
Helen

PYTHON LISTS

Lists can store elements of any data type, including other lists.

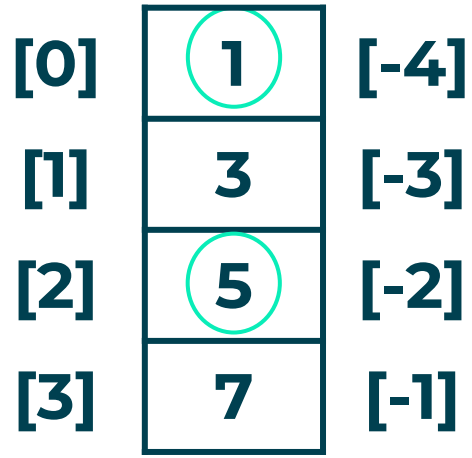
```
mix = [1,3.14,"fruit",True]
```

1
3.14
“fruit”
True

ADDRESSING AN ELEMENT USING ITS INDEX

```
numbers = [1,3,5,7]

print(numbers[0])
print(numbers[2])
print(numbers[-2])
```



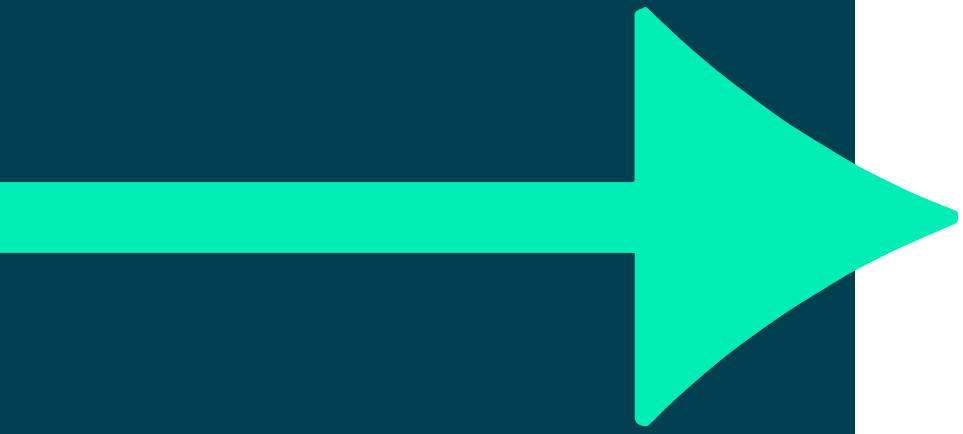

```
names = ["Bob", "Steve", "Helen"]

print(names[1])
Print(names[-1])
```

Applies to strings (lists of characters) too

```
s = ‘Hello world!’
print(s[1])
print(s[-5])
```

ADDRESSING MULTIPLE ELEMENTS - SLICING



identifier[start:stop:step]

- ***stop*** is the position after the last index
- Default ***start*** is 0
- Default ***stop*** is the last index
- Default ***step*** is 1

```
numbers = [1,3,5,7,9,11,13,15,17,19,21]

print(numbers[3:10])
print(numbers[2:])
print(numbers[:6])
Print(numbers[::2])
```

```
Applies to strings (lists of characters) too
s = 'Hello world!'
print(s[3:10])
print(s[:6])
Print(s[::-1])
```

THE LEN() FUNCTION

Use the len() function to Get the length of a list

```
numbers = [1,3,5,7]

print(len(numbers))
```

4

```
names = ["Bob","Steve","Helen"]

print(len(names))
```

3

ITERATING THROUGH A LIST - FOR LOOP

```
names = ["Bob", "Steve", "Helen"]

for name in names:
    print(name)
```

```
Bob
Steve
Helen

Press any key to continue . . .
```

ITERATING THROUGH A LIST – WHILE LOOP

```
names = ["Bob", "Steve", "Helen"]

i=0      # used as index

while(i < len(names) ):
    print(names[i])
    i += 1
```

```
Bob
Steve
Helen

Press any key to continue . . .
```

APPENDING ELEMENTS AT THE END

```
numbers = [1,3,5,7,9]

numbers.append(88)
```

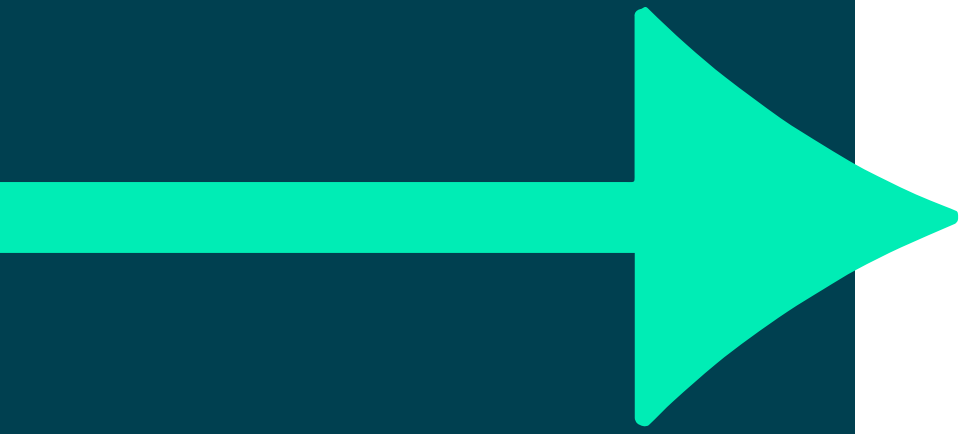
[1, 3, 5, 7, 9, 88]

```
You can start with an empty List:

nos = []
nos.append(10)
nos.append(20)
nos.append(30)
```

[10, 20, 30]

INSERTING AN ELEMENT AT A SPECIFIED POSITION



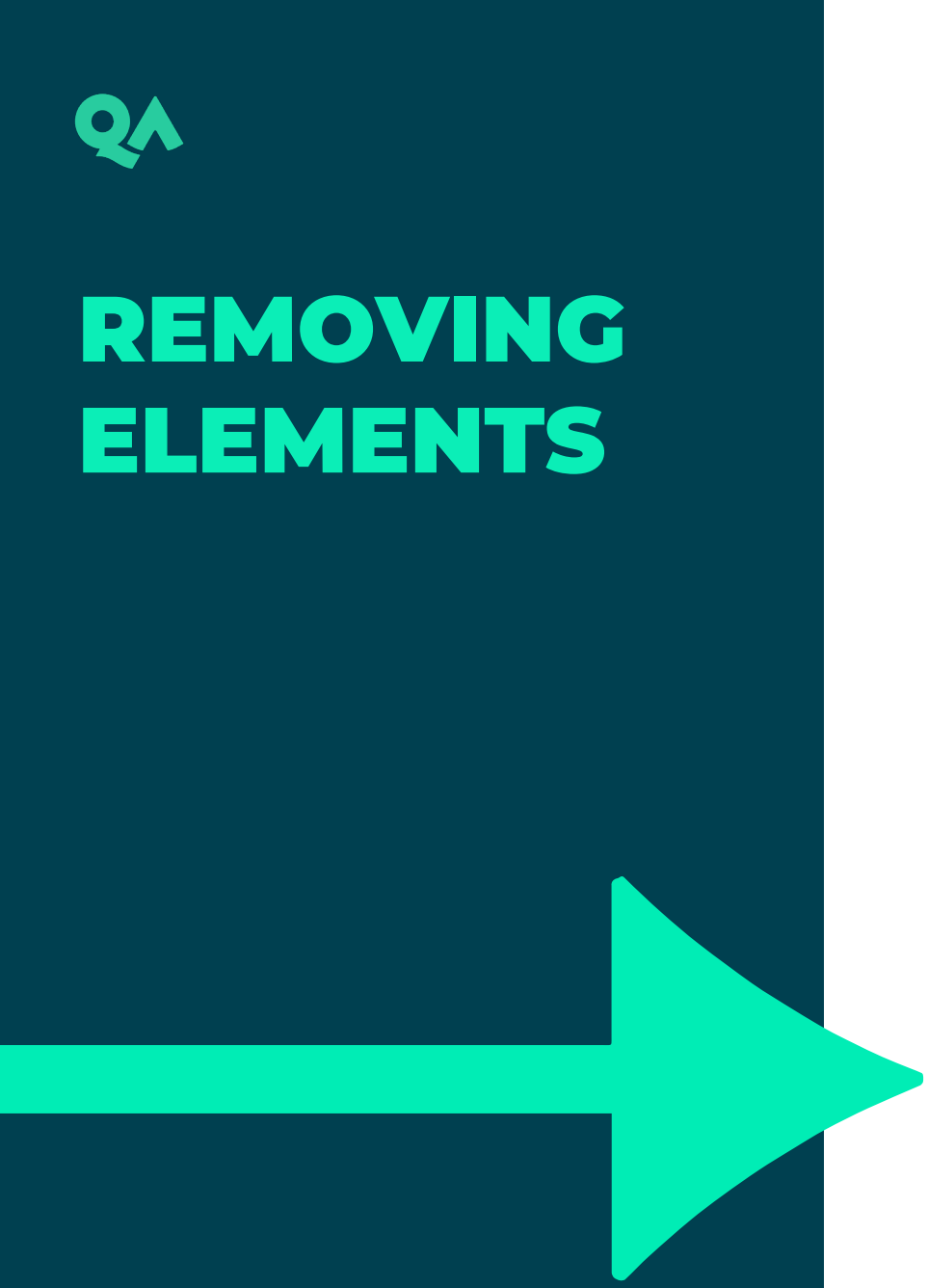
Insert 88 at position (index) 1.

Don’t forget that in Python the first index is 0.

```
numbers = [1,3,5,7,9]

numbers.insert(1,88)
```

[1, 88, 3, 5, 7, 9]



REMOVING ELEMENTS

```
numbers = [1,3,5,7,5,9,5]

numbers.remove(5)
```

[1, 3, 7, 5, 9, 5]

```
del(numbers[0])
```

[3, 7, 5, 9, 5]

```
names = ['Bob', 'Steve', 'Helen']

names.remove("Steve")
```

['Bob', 'Helen']

Works with any kind of list

CHANGING THE VALUE OF ELEMENTS

```
numbers = [1,3,5,7,5,9,5]

numbers[2] = 999
```

[1, 3, 999, 7, 5, 9, 5]

```
names = ['Bob', 'Steve', 'Helen’]

names[2] = “Chris"
```

['Bob’, ‘Chris’, 'Helen']

CHECKING FOR EXISTENCE

Use the **in** command to check if an item is present in a list

```
adminIDs = [12,33,84,45,67,36,16,66,67,99]
id = int(input('Enter your ID '))
if id in adminIDs:
    print('Welcome!')
```

numeric

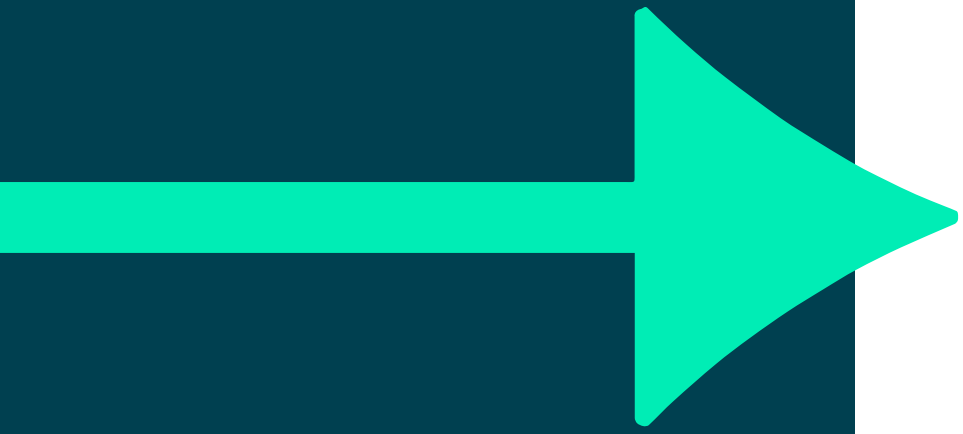
```
names = ['David','John','Joanne','Sean','Sonia']
if 'Sean' in names:
    print('Sean is in the list!')
```

strings

```
names = "David,John,Joanne,Sean,Sonia"
if 'John' in names:
    print('John is in the list!')
```

List of characters
(string)

SORTING LIST ELEMENTS



Use the **sort**() function to sort elements

```
ages = [12,33,84,45,67,36,16]

ages.sort()
print(ages)
```

[12, 16, 33, 36, 45, 67, 84]

```
ages = [12,33,84,45,67,36,16]

ages.sort(reverse=True)
print(ages)
```

[84, 67, 45, 36, 33, 16, 12]

SORTING LIST ELEMENTS

Beware when sorting list elements which are strings:

```
c = ['Apple Inc', 'apple', 'pear', 'tomato', 'bear', 'wolf', 'Zebra']
c.sort()
c
```

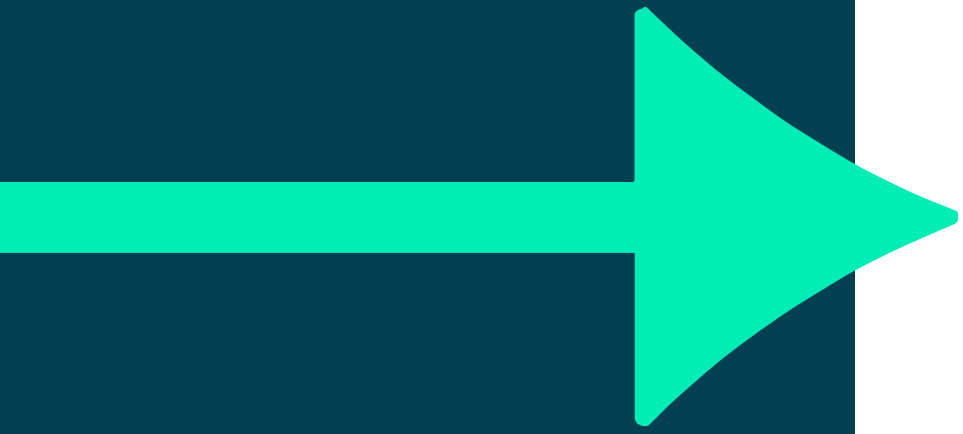
```
['Apple Inc', 'Zebra', 'apple', 'bear', 'pear', 'tomato', 'wolf']
```

How strange! Why is Zebra before apple?

By default, the sort() method sorts the list in ASCII order rather than actual alphabetical order. This means uppercase letters come before lowercase letters.

To sort the values in regular alphabetical order, set key to str.lower or str.upper. This causes the sort() function to treat all the list items as if they were lowercase/uppercase without actually changing the values in the list.

SORTING LIST ELEMENTS



Beware when sorting list elements which are strings:

```
c = ['Apple Inc', 'apple', 'pear', 'tomato', 'bear', 'wolf', 'Zebra']
c.sort(key=str.upper)
c
```

```
['apple', 'Apple Inc', 'bear', 'pear', 'tomato', 'wolf', 'Zebra']
```

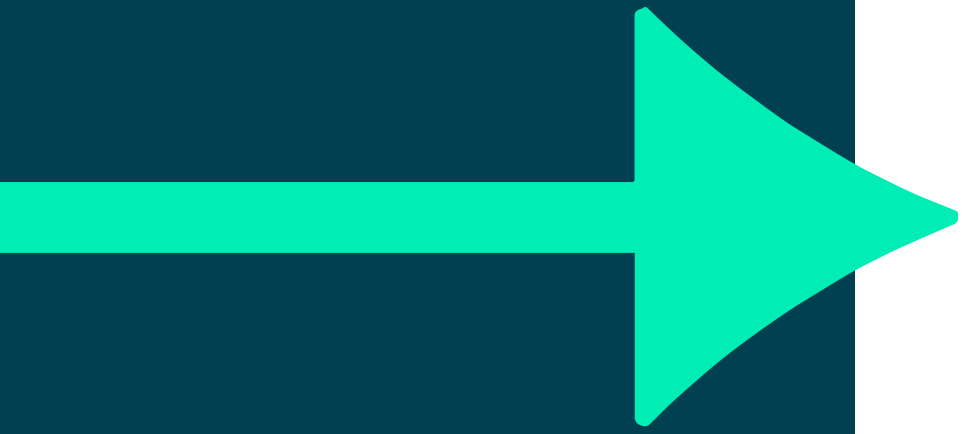
```
c.sort(key=str.upper, reverse=True)
c
```

```
['Zebra', 'wolf', 'tomato', 'pear', 'bear', 'Apple Inc', 'apple']
```

LISTS OF LISTS

```
# list of lists
prod1 = ["a123","Hammer",2.75]
prod2 = ["a124","Screwdriver",0.95]
prod3 = ["a125","Pliers",1.62]
products = [prod1, prod2, prod3]
print(products)
```

```
[['a123', 'Hammer', 2.75], ['a124', 'Screwdriver', 0.95], ['a125', 'Pliers', 1.62]]
```



```
for prod in products:
    print(prod[0],"\t",prod[1],"\t",prod[2])
```

```
a123    Hammer          2.75
a124    Screwdriver     0.95
a125    Pliers          1.62
Press any key to continue . . .
```

STRINGS (LISTS OF CHARACTERS)

Strings can be treated as lists of characters, so functionality related to lists apply to them too.

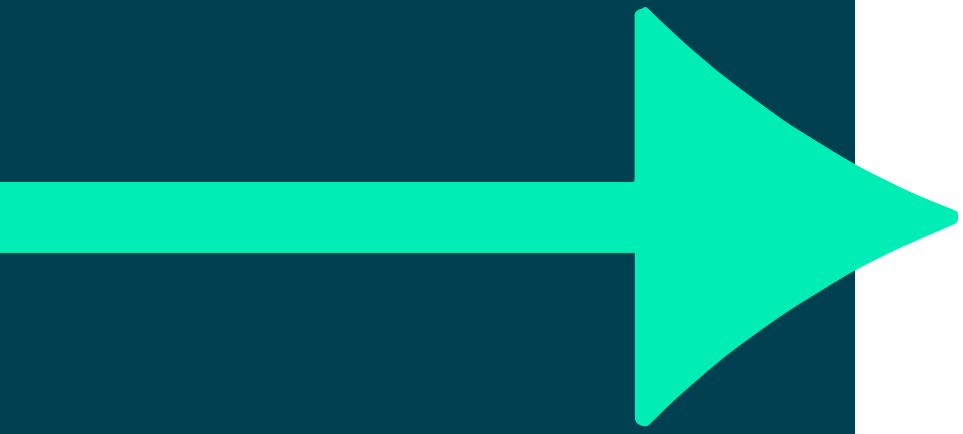
```
greeting = "Hello"
for x in greeting:
    print(x)
```

```
H
e
l
l
o
Press any key to continue . . .
```

```
greeting = "Hello"
i=0      # used as index

while(i < len(greeting)):
    print(greeting[i])
    i += 1
```

STRINGS (LISTS OF CHARACTERS)



Major difference

- Lists are mutable
- Strings are immutable

We can change the value of an element of a list, but not of an element of a string:

```
L = [1, 2, 3, 4, 5]
L[4] = 0
L
```

```
[1, 2, 3, 4, 0]
```

```
S = '12345'
S[4] = 0
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-9-f463423e9604> in <module>
      1 S = '12345'
----> 2 S[4] = 0

TypeError: 'str' object does not support item assignment
```

STRING METHODS

Python has a set of built-in methods that can be used on strings.

Note: All string methods returns new values. They do not change the original string.

```
test = 'britain'

test.capitalize()
test
```

```
'britain'
```

```
test1 = test.capitalize()
test1
```

```
'Britain'
```

A list of Python string methods is available here:

https://www.w3schools.com/python/python_ref_string.asp

STRING METHODS

Some examples of string methods:

```
testGB = 'great britain'

# Convert the first character to upper case
testGB1 = testGB.capitalize()
testGB1
```

```
'Great britain'
```

```
# Convert the first character of each word to upper case
testGB2 = testGB.title()
testGB2
```

```
'Great Britain'
```

```
# Convert a string into lower case
testGB3 = testGB.lower()
testGB3
```

```
'great britain'
```

STRING METHODS - CONTINUED

Some examples of string methods:

```
# Convert a string into upper case
testGB4 = testGB.upper()
testGB4
```

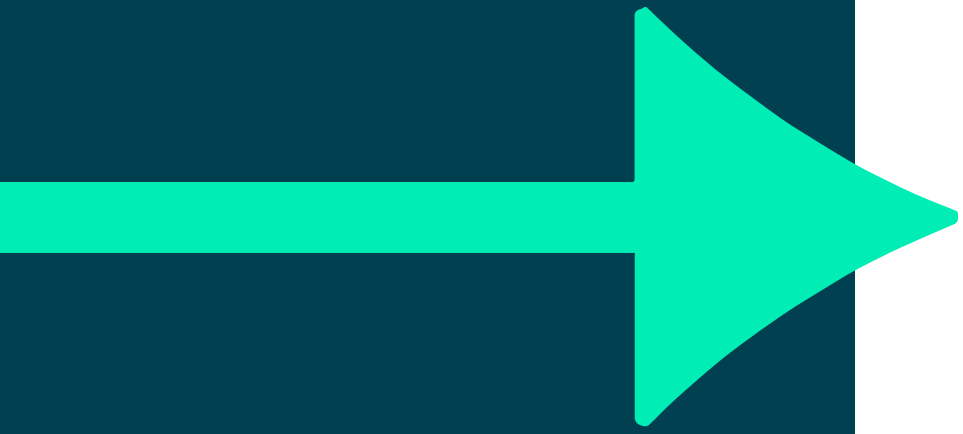
```
'GREAT BRITAIN'
```

```
# Search the string for a specified value and return the position of where it was found
testGB5 = testGB.find('BRITAIN')
testGB5
```

```
-1
```

```
# Python is case sensitive!
testGB5_1 = testGB.find('britain')
testGB5_1
```

STRING METHODS - CONTINUED



Some examples of string methods:

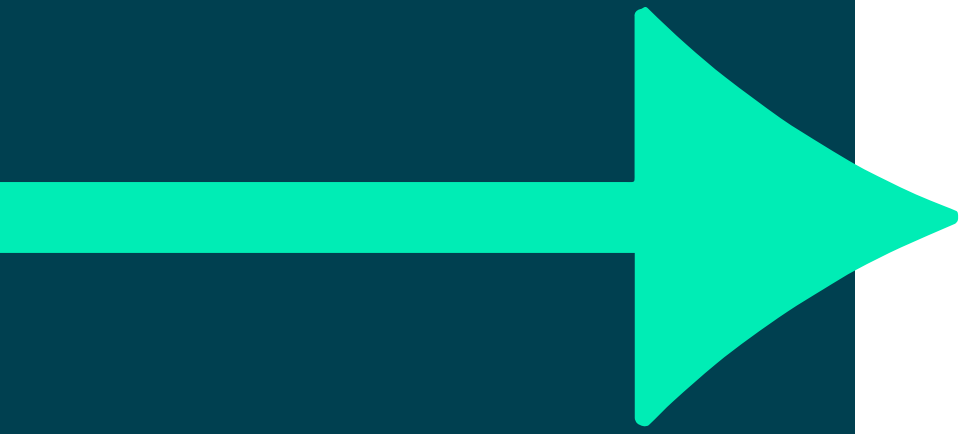
```
# Return True if all characters in the string are in the alphabet
testGB6 = testGB.isalpha()
testGB6
```

```
False
```

```
# Return True if the string starts with the specified value
testGB6 = testGB.startswith('g')
testGB6
```

```
True
```

THE STRING.SPLIT() METHOD



Used for splitting and extracting elements from a String using a Delimiter

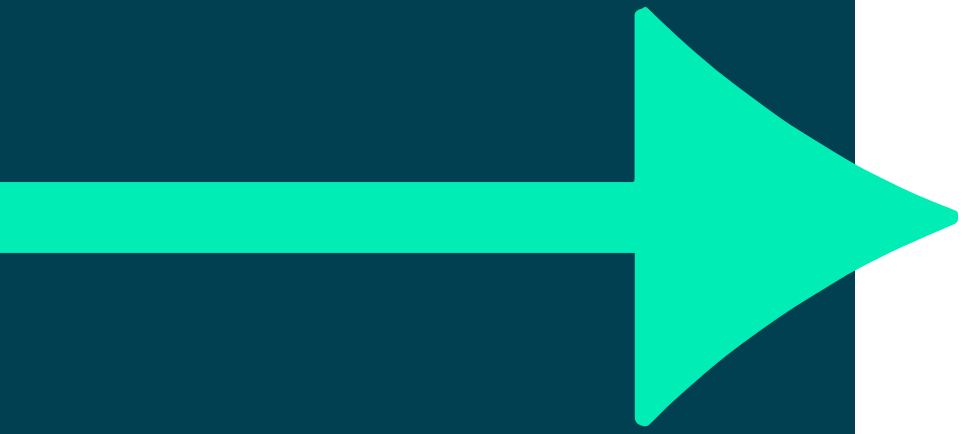
```
data = 'Bob,Steve,Helen'
names = data.split(',')
print(names)
```

['Bob', 'Steve', 'Helen']

```
data='18/OCT/2020'
parts = data.split('/')
print(parts)
```

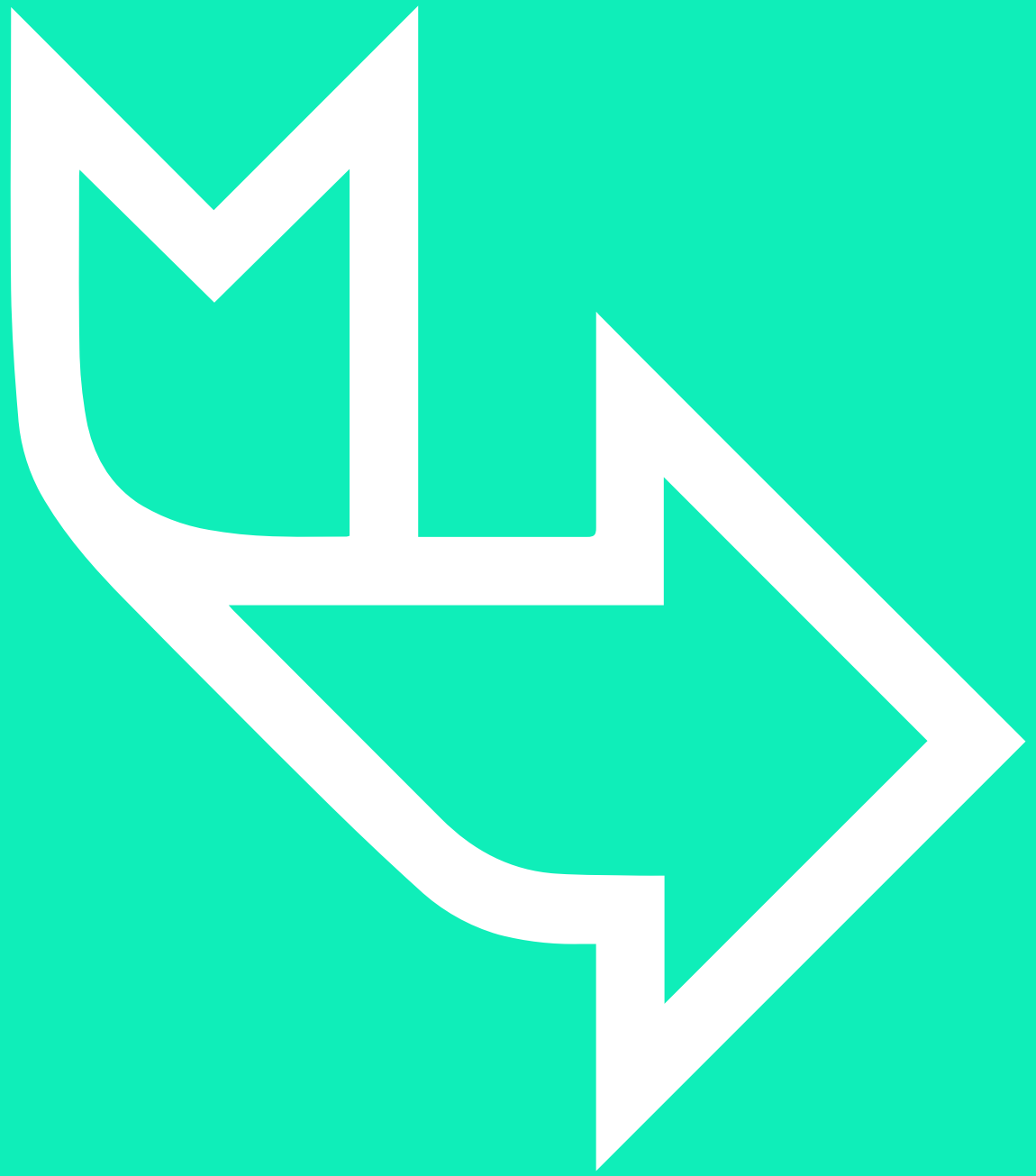
['18', 'OCT', '2020']

SUMMARY



In this lesson, you've learned how to:

- Create a list
- Read an item from a list
- Add an item to a list
- Remove an item from a list
- Replace an item
- Strings as lists of characters
- String methods



Further Reading

https://www.python.org/