Strings

CC BY

String Data Type

- A string is a sequence of characters
- A string literal uses quotes 'Hello' or "Hello"
- For strings, + means “concatenate”
- When a string contains numbers, it is still a string
- We can convert numbers in a string into a number using int()

```
>>> str1 = "Hello"
>>> str2 = 'there'
>>> bob = str1 + str2
>>> print(bob)
Hellothere
>>> str3 = '123'
>>> str3 = str3 + 1
Traceback (most recent call
last):  File "<stdin>", line 1,
in <module>
TypeError: cannot concatenate
'str' and 'int' objects
>>> x = int(str3) + 1
>>> print(x)
124
>>>
```

Looking Inside Strings

- We can get at any single character in a string using an index specified in square brackets

- The index value must be an integer and starts at zero

- The index value can be an expression that is computed

b	a	n	a	n	a
0	1	2	3	4	5

```
>>> fruit = 'banana'
>>> letter = fruit[1]
>>> print(letter)
a
>>> x = 3
>>> w = fruit[x - 1]
>>> print(w)
n
```

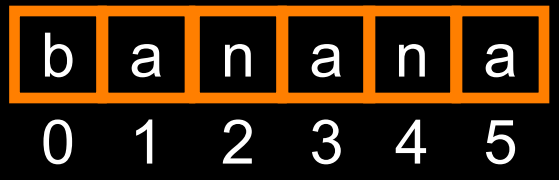
A Character Too Far

- You will get a python error if you attempt to index beyond the end of a string
- So be careful when constructing index values and slices

```
>>> zot = 'abc'
>>> print(zot[5])
Traceback (most recent call last):  File "<stdin>", line 1, in <module>
IndexError: string index out of range
>>>
```

Strings Have Length

The built-in function len gives us the length of a string

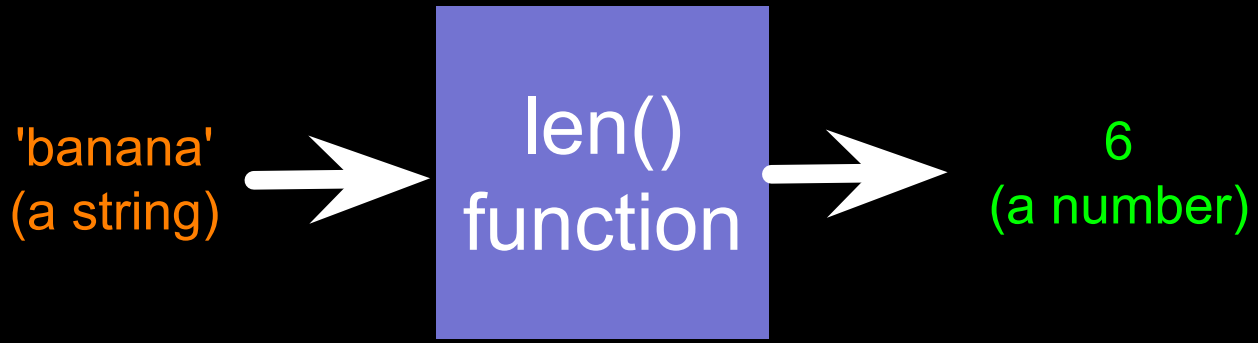



```
>>> fruit = 'banana'
>>> print(len(fruit))
6
```

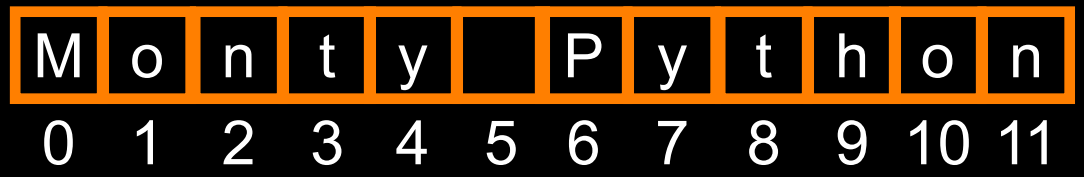
len Function

```
>>> fruit = 'banana'
>>> x = len(fruit)
>>> print(x)
6
```

A function is some stored code that we use. A function takes some input and produces an output.



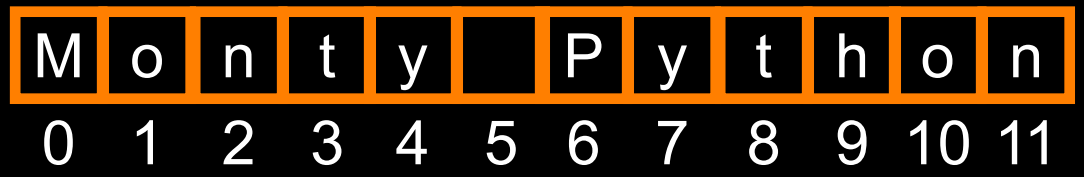
Slicing Strings



- We can also look at any continuous section of a string using a colon operator
- The second number is one beyond the end of the slice - “up to but not including”
- If the second number is beyond the end of the string, it stops at the end

```
>>> s = 'Monty Python'
>>> print(s[0:4])
Mont
>>> print(s[6:7])
P
>>> print(s[6:20])
Python
```

Slicing Strings



If we leave off the first number or the last number of the slice, it is assumed to be the beginning or end of the string respectively

```
>>> s = 'Monty Python'
>>> print(s[:2])
Mo
>>> print(s[8:])
thon
>>> print(s[:])
Monty Python
```

What is the output of:

`'Chicago'[3:7]`?

String Concatenation

When the + operator is applied to strings, it means "concatenation"

```
>>> a = 'Hello'
>>> b = a + 'There'
>>> print(b)
HelloThere
>>> c = a + ' ' + 'There'
>>> print(c)
Hello There
>>>
```

Using in as a Logical Operator

- The in keyword can also be used to check to see if one string is “in” another string
- The in expression is a logical expression that returns True or False

```
>>> fruit = 'banana'
>>> 'n' in fruit
True
>>> 'm' in fruit
False
>>> 'nan' in fruit
True
```

String Library

- Python has a number of string functions which are in the string library
- These functions are already built into every string - we invoke them by appending the function to the string variable
- These functions do not modify the original string, instead they return a new string that has been altered

```
>>> greet = 'Hello Bob'
>>> zap = greet.lower()
>>> print(zap)
hello bob
>>> print(greet)
Hello Bob
>>> print('Hi There'.lower())
hi there
>>>
```

Making everything UPPER CASE

- You can make a copy of a string in lower case or upper case
- Often when we are searching for a string using find() we first convert the string to lower case so we can search a string regardless of case

```
>>> greet = 'Hello Bob'
>>> nnn = greet.upper()
>>> print(nnn)
HELLO BOB
>>> www = greet.lower()
>>> print(www)
hello bob
>>>
```

Search and Replace

- The replace() function is like a “search and replace” operation in a word processor
- It replaces all occurrences of the search string with the replacement string

```
>>> greet = 'Hello Bob'
>>> nstr = greet.replace('Bob','Jane')
>>> print(nstr)
Hello Jane
>>> nstr = greet.replace('o','X')
>>> print(nstr)
HellX BXb
>>>
```

Stripping Whitespace

- Sometimes we want to take a string and remove whitespace at the beginning and/or end

- lstrip() and rstrip() remove whitespace at the left or right

- strip() removes both beginning and ending whitespace

```
>>> greet = '   Hello Bob  '
>>> greet.lstrip()
'Hello Bob  '
>>> greet.rstrip()
'   Hello Bob'
>>> greet.strip()
'Hello Bob'
>>>
```

Finding Occurrences

- The index() function finds the position of the first occurrence (or ***index***) of a character in a string

- If the character does not exist in the string, an error is thrown

```
>>> greet = 'Hello Bob'
>>> greet.index('B')
6
>>> greet.index(' ')
5
>>> greet.index('l')
2
>>>
```

What is the output of:

```
birds = 'Chicago'.upper().replace('AGO', 'ks')
print('cks' in bird)
```

Review (True or False)

1. Strings in Python are 0-based indexed (i.e., the first character is at index 0).
2. In Python, you can directly modify a character within a string by using the index notation, like `s[2] = "z"`.
3. Characters are a different type from strings in Python

Review (True or False)

1. Strings in Python are 0-based indexed (i.e., the first character is at index 0).
2. In Python, you can directly modify a character within a string by using the index notation, like `s[2] = "z"`.
3. Characters are a different type from strings in Python

Python Lists

Lists

- A list allows us to put many values in a single variable

```
friends = [ 'Joseph', 'Glenn', 'Sally' ]
```

List Constants

- List constants are surrounded by square brackets and the elements in the list are separated by commas
- A list element can be any Python object - even another list
- A list can be empty

```
>>> print([1, 24, 76])
[1, 24, 76]
>>> print(['red', 'yellow',
'blue'])
['red', 'yellow', 'blue']
>>> print(['red', 24, 98.6])
['red', 24, 98.6]
>>> print([ 1, [5, 6], 7])
[1, [5, 6], 7]
>>> print([])
[]
```

Looking Inside Lists

Just like strings, we can get at any single element in a list using an index specified in square brackets


```
>>> friends = [ 'Joseph', 'Glenn', 'Sally'
]
>>> print(friends[1])
Glenn
>>>
```

Lists are Mutable

- Strings are “immutable” - we cannot change the contents of a string - we must make a new string to make any change
- Lists are “mutable” - we can change an element of a list using the index operator

```
>>> fruit = 'Banana'
>>> fruit[0] = 'b'
Traceback
TypeError: 'str' object does not support item assignment
>>> x = fruit.lower()
>>> print(x)
banana
>>> lotto = [2, 14, 26, 41, 63]
>>> print(lotto)
[2, 14, 26, 41, 63]
>>> lotto[2] = 28
>>> print(lotto)
[2, 14, 28, 41, 63]
```

How Long is a List?

- The len() function takes a list as a parameter and returns the number of elements in the list

```
>>> x = [ 1, 2, 'joe', 99]
>>> print(len(x))
4
```

How Long is a List?

- The len() function takes a list as a parameter and returns the number of elements in the list
- Actually len() tells us the number of elements of any set or sequence (such as a string...)

```
>>> x = [ 1, 2, 'joe', 99]
>>> print(len(x))
4
>>> greet = 'Hello Bob'
>>> print(len(greet))
9
>>>
```

Concatenating Lists Using +

We can create a new list by adding two existing lists together

```
>>> a = [1, 2, 3]
>>> b = [4, 5, 6]
>>> c = a + b
>>> print(c)
[1, 2, 3, 4, 5, 6]
>>> print(a)
[1, 2, 3]
```

Lists Can Be Sliced Using :

```
>>> t = [9, 41, 12, 3, 74, 15]
>>> t[1:3]
[41,12]
>>> t[:4]
[9, 41, 12, 3]
>>> t[3:]
[3, 74, 15]
>>> t[:]
[9, 41, 12, 3, 74, 15]
```

Remember: Just like in strings, the second number is “up to but not including”

What is the output of:

```
myList = ['Hotdog', 'Pizza', 'Ice Cream', 'Soda']
print(myList[:-2])
```

Is Something in a List?

- Python provides the operator in, which lets you check if an item is in a list
- It is a logical operator that returns True or False
- Using not will flip the True/False value of in
- They do not modify the list

```
>>> some = [1, 9, 21, 10, 16]
>>> 9 in some
True
>>> 15 in some
False
>>> 20 not in some
True
>>>
```

Counting Occurrences in a List

- We can count the number of occurrences in a empty list using the count method

```
>>> stuff = ['book', 'movie', 'book']
>>> stuff.count('book')
>>> 2
```

Building a List from Scratch

- We can create an empty list and then add elements using the append method
- The list stays in order and new elements are added at the end of the list

```
>>> stuff = [1, 2, 3]
stuff = list()
>>> stuff.append('book')
>>> stuff.append(99)
>>> print(stuff)
['book', 99]
>>> stuff.append('cookie')
>>> print(stuff)
['book', 99, 'cookie']
```

What is the output of:

```
school = ["nemo", "marlin", "dory"]
school2 = school
school2.append("gill")
print(school)
```

Lists are in Order

- A list can hold many items and keeps those items in the order until we do something to change the order
- A list can be sorted (i.e., change its order)

```
>>> friends = [ 'Joseph', 'Glenn', 'Sally' ]
>>> friends.sort()
>>> print(friends)
['Glenn', 'Joseph', 'Sally']
>>> print(friends[1])
Joseph
>>>
```

Converting a String to a List

- We can convert a string to a list by using the string split method
- This method will split the string on the character or short string provided

```
>>> word = 'words*separated*by*stars'
>>> word.split('*')
>>> ['words', 'separated', 'by', 'stars']
```

Built-in Functions and Lists

- There are a number of functions built into Python that take lists as parameters

```
>>> nums = [3, 41, 12, 9, 74, 15]
>>> print(len(nums))
6
>>> print(max(nums))
74
>>> print(min(nums))
3
>>> print(sum(nums))
154
>>> print(sum(nums)/len(nums))
25.6
```

Review (True or False)

1. The `append()` method adds an element to the beginning of a list in Python
2. It is possible to have the following list:
   `['string', 1, 0.5, True]`
3. Lists have a fixed maximum size, and once this size is reached, no more elements can be added to the list.

Review (True or False)

1. The `append()` method adds an element to the beginning of a list in Python
2. It is possible to have the following list:
   `['string', 1, 0.5, True]`
3. Lists have a fixed maximum size, and once this size is reached, no more elements can be added to the list.

Acknowledgements / Contributions

These slides are Copyright 2010- Charles R. Severance (www.dr-chuck.com) of the University of Michigan School of Information and open.umich.edu and made available under a Creative Commons Attribution 4.0 License. Please maintain this last slide in all copies of the document to comply with the attribution requirements of the license. If you make a change, feel free to add your name and organization to the list of contributors on this page as you republish the materials.

Initial Development: Charles Severance, University of Michigan School of Information

Adapted by Minh Tran, University of Chicago, Department of Computer Science for use in the pre-college course Creating Computer Games for Learning (Summer 2026).

...