



THE UNIVERSITY OF
WESTERN
AUSTRALIA

Lecture 10

Strings

Objectives

- To understand the string data type and how strings are represented in a computer.
- To understand the basic idea of sequences and indexing as they apply to Python strings and lists.

The String Data Type

- The most common use of personal computers is word processing.
- Text is represented in programs by the string data type.
- A string is a sequence of characters enclosed within quotation marks (") or apostrophes (').

The String Data Type

```
>>> str1="Hello"  
>>> str2='spam'  
>>> print(str1, str2)  
Hello spam  
>>> type(str1)  
<class 'str'>  
>>> type(str2)  
<class 'str'>
```

The String Data Type

- Getting a string as input

```
>>> firstName = input("Please enter your name: ")
```

```
Please enter your name: John
```

```
>>> print("Hello", firstName)
```

```
Hello John
```

- Notice that the input is not evaluated. We want to store the typed characters, not to evaluate them as a Python expression, e.g. convert to `int`.

The String Data Type

- We can access the individual characters in a string through **indexing**.
- The positions in a string are numbered from the left, starting with **0**.
- The general form is `<string> [<expr>]` where the value of `expr` (i.e. an integer) determines which character is selected from the string.

The String Data Type

H	e	l	l	o		B	o	b
0	1	2	3	4	5	6	7	8

```
>>> greet = "Hello Bob"
```

```
>>> greet[0]
```

```
'H'
```

```
>>> print(greet[0], greet[2], greet[4])
```

```
H l o
```

```
>>> x = 8
```

```
>>> print(greet[x - 2])
```

```
B
```

The String Data Type

H	e	l	l	o		B	o	b
0	1	2	3	4	5	6	7	8

- In a string of n characters, the last character is at position $n-1$ since we start counting with 0.
- We can index from the right side using negative indexes.

```
>>> greet[-1]
```

```
'b'
```

```
>>> greet[-3]
```

```
'B'
```


The String Data Type

- Indexing returns a string containing a single character from a larger string.
- We can also access a contiguous sequence of characters, called a **substring**, through a process called **slicing**.

Slicing Strings



changhsinlee.com

The String Data Type

- Slicing:
`<string> [<start> : <end>]`
- start and end should both be *ints*
- The slice contains the substring beginning at position start and runs up to **but does NOT include** the position end.

The String Data Type

H	e	l	l	o		B	o	b
0	1	2	3	4	5	6	7	8

```
>>> greet[0:3]
```

```
'Hel'
```

```
>>> greet[5:9]
```

```
' Bob'
```

```
>>> greet[:5]
```

```
'Hello'
```

```
>>> greet[5:]
```

```
' Bob'
```

```
>>> greet[:]
```

This is same as greet

```
'Hello Bob'
```

The String Data Type

- If either **start** or **end** expression is missing, then the start or the end of the string are used.
- Can we put two strings together into a longer string?
- **Concatenation** “glues” two strings together (+)
- **Repetition** builds up a string by multiple concatenations of a string with itself (*)

The String Data Type

Operator	Meaning
+	Concatenation
*	Repetition
<string>[]	Indexing
<string>[:]	Slicing
len(<string>)	Length
for <var> in <string> : Iteration through characters	

'spameggs'

' SpamAndEggs '

'spamspamspam'

'spamspamspamspamspam'

```
'spamspamsameggseggseggseggseggsegg'
```

The String Data Type

The function `len()` is used to return the length of string.

```
>>> len("spam")
```

```
4
```

```
>>> for ch in "Spam!":  
    print(ch, end=" ")
```

Note: `\` `'` printed after
each character value

```
S p a m !
```


Summary

- We learned how strings are represented in a computer.
- We learned about substrings and string slicing.
- We learned how various operations can be performed on strings.