香港浸會大學
HONG KONG BAPTIST UNIVERSITY

Strings

JOUR7280/COMM7780

Big Data Analytics for Media and Communication

Instructor: Dr. Xiaoyi Fu

String Data Type

- A string is a sequence of characters
- A string 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()

```
In [20]: sval = '123'
         type(sval)

Out[20]: str

In [21]: print(sval+1)

---------------------------------------------------------------------------
TypeError                                 Trace
back (most recent call last)
<ipython-input-21-d31b14f87b22> in <module>
----> 1 print(sval+1)

TypeError: can only concatenate str (not "int")
to str

In [22]: ival = int(sval)
         type(ival)

Out[22]: int

In [23]: print(ival+1)

         124

In [24]: nsv = 'hello world'
         niv = int(nsv)

---------------------------------------------------------------------------
ValueError                                Trace
back (most recent call last)
<ipython-input-24-7b19be68013f> in <module>
      1 nsv = 'hello world'
----> 2 niv = int(nsv)

ValueError: invalid literal for int() with base
10: 'hello world'
```

Read & Convert

- We prefer to read data in using strings then parse and convert data as we need
- This gives us more control over error situation and/or bad user input
- Input numbers must be converted from string

```
In [*]: name = input('Who are u?')
        print('Welcome', name)

        Who are u?
```

```
In [27]: name = input('Who are u?')
         print('Welcome', name)

         Who are u?xiaoyi
         Welcome xiaoyi
```

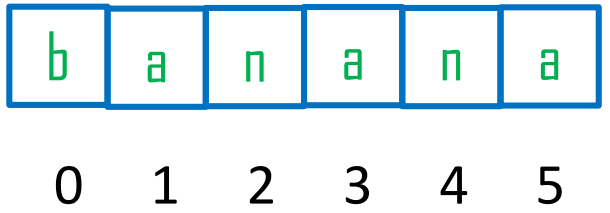
```
# convert elevator floors
inp = input('Europe floor?')
usf = int(inp)+1
print('US Floor', usf)
```

Europe floor? 0

US Floor 1

Look Inside Strings

- We can get any single character in a string using an index specified in square brackets
- The index value must be an integer and starts at 0
- The index value can be an expression that is computed




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

```
a
n
```

6 strings.ipynb

A Character Too Far

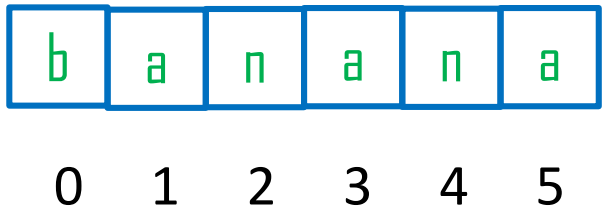
- You will get a python error if you attempt to index beyond the end of a string
- Be careful when constructing index values and slices

```
In [4]:
```

```
zot = 'abc'
print(zot[5])
```

```
---------------------------------
---------------------------------
-
IndexError                                Trace
back (most recent call last)
<ipython-input-4-6acb855f1a9c> in <mo
dule>
      1 zot = 'abc'
----> 2 print(zot[5])

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
```

Loop and Count

- This is a simple loop that loops through each letter in a string and counts the number of times the loop encounters the 'a' character

```
word = 'banana'
count = 0
for char in word:
    if char == 'a':
        count = count + 1
print(count)
```

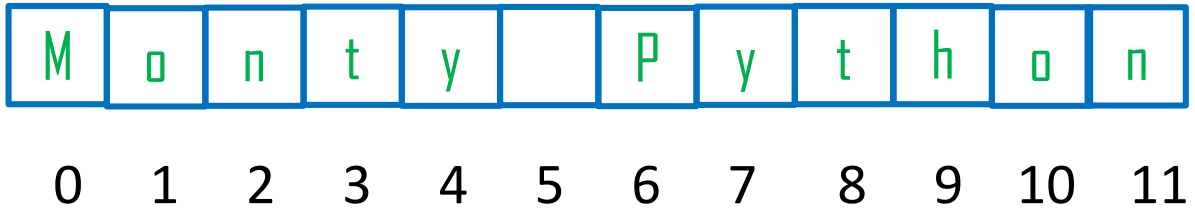
Output:

3

6 strings.ipynb

String Slices

- We can also look at any continuous section of a string using a colon operator
- The operator returns the part of the string from the "n-th" character to the "m-th" character,
  - Including the first but excluding the last.
- The second number is one beyond the end of the slice
  - Up to but not including
- If the second number is beyond the string, it stops at the end

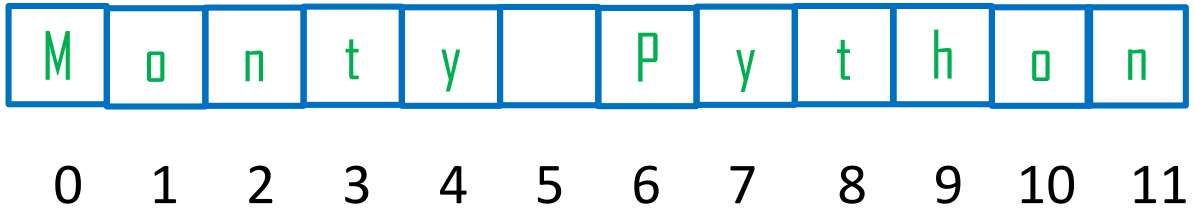



```
s = 'Monty Python'
print(s[0:4])
print(s[6:7])
print(s[6:20])
```

```
Mont
P
Python
```

6 strings.ipynb

String Slices



- If you omit the first index (before the colon), the slice starts at the beginning of the string.
- If you omit the second index, the slice goes to the end of the string

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

```
Mo
thon
Monty Python
```

6 strings.ipynb

String Concatenation

```
a = 'Hello'
b = a + 'there'
print(b)
c = a + ' ' + 'there'
print(c)
```

```
Hellothere
Hello there
```

- When the + applied to strings, it means concatenation

6 strings.ipynb

Use in as a Logical Operator

- The in keyword can also be used to check if one string is "in" another string
- The in expression is a logical expression that returns True or False and can be used in an if statement

```
fruit = 'banana'
print('n' in fruit)
print('m' in fruit)
print('nan' in fruit)
if 'a' in fruit:
    print('Found it!')
```

```
True
False
True
Found it!
```

6 strings.ipynb

String Comparison

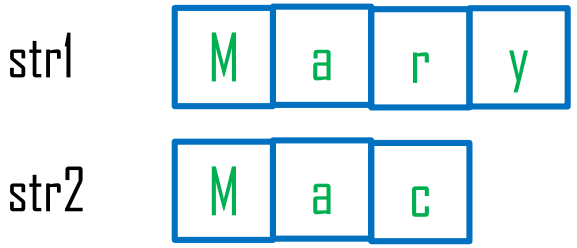
```
if word == 'banana‘:
    print('All right, bananas.')

if word < 'banana‘:
    print('Your word,' + word + ', comes before banana.')
elif word > 'banana‘:
    print('Your word,' + word + ', comes after banana.')
else:
    print('All right, bananas.')
```

- Other comparison operations are useful for putting words in alphabetical order
- Python does not handle uppercase and lowercase letters the same way that people do.
  - All the uppercase letters come before all the lowercase letters

6 strings.ipynb

String Comparison

- Suppose you have str1 as "Mary" and str2 as "Mac".
- The first two characters from str1 and str2 (M and M) are compared.
- As they are equal, the second two characters are compared.
- Because they are also equal, the third two characters (r and c) are compared.
- And because r has greater ASCII value than c, so str1 > str2.




```
>>> "tim" == "tie"
False
>>> "free" != "freedom"
True
>>> "arrow" > "aron"
True
>>> "right" >= "left"
True
>>> "teeth" < "tee"
False
>>> "yellow" <= "fellow"
False
>>> "abc" > ""
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.

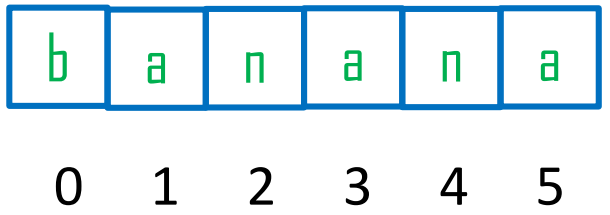
```
greet = 'Hello Tony'
zap = greet.lower()
print(zap)
print(greet)
print('Hi There'.lower())
```

```
hello tony
Hello Tony
hi there
```

6 data structures.ipynb

Search a String

- We use find() function to search for the position of one string within another
- find() finds the first occurrence of the substring
- If the substring is not found, find() returns -1
- Remember that string position starts at zero




```
fruit = 'banana'
pos = fruit.find('na')
print(pos)
aa = fruit.find('z')
print(aa)
```

```
2
-1
```

6 strings.ipynb

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 Tony'
nstr = greet.replace('Tony', 'Peter')
print(nstr)
nstr = greet.replace('o', 'X')
print(nstr)
```

```
Hello Peter
HellX TXny
```

6 strings.ipynb

Strip 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 whitespaces

```
greet = '    Hello Tony    '
print(greet.lstrip())
print(greet.rstrip())
print(greet.strip())
print(greet)
```

```
Hello Tony
    Hello Tony
Hello Tony
    Hello Tony
```

6 strings.ipynb

Prefixes

```
line = 'Have a nice day'
print(line.startswith('Have'))
print(line.startswith('p'))
print(line.startswith('h'))
```

```
True
False
False
```

```
line = 'Have a nice day'
print(line.startswith('h'))
print(line.lower().startswith('h'))
```

```
False
True
```

- startswith() requires case to match, so sometimes we take a line and map it all to lowercase before we do any checking using the lower method.

6 strings.ipynb

Parsing Strings

```
data = 'From stephen.marquard@uct.ac.za Sat Jan 5 09:14:16 2008'
atpos = data.find('@')
print(atpos)
sppos = data.find(' ',atpos)
print(sppos)
host = data[atpos+1:sppos]
print(host)
```

```
21
31
uct.ac.za
```

- We use a version of the find method which allows us to specify a position in the string where we want find to start looking.
- When we slice, we extract the characters from "one beyond the at-sign through up to but not including the space character".

6 strings.ipynb

Acknowledgements / Contributions

- Some of the slides used in this lecture from:
  - Charles R. Severance - University of Michigan School of Information

Thank You