Chapter 9

Dr. Raza Ul Mustafa Khokhar
American University
Computer Science Department - CSC-148

Strings

My_var = “Hellow world!”

Strings operations

- You cannot perform mathematical operations on strings
- However, + represents concatenation

```
message - 1
"Hello" / 123
message * "Hello"
"15" + 2
```

```
fruit = "banana"
bakedGood = " nut bread"
print(fruit + bakedGood)
```

Look at this example!

```python
print("Go" * 6)

name = "Packers"
print(name * 3)

print(name + "Go" * 3)

print((name + "Go") * 3)
```

Index operator

```
school = "Luther College"
m = school[2]
print(m)

lastchar = school[-1]
print(lastchar)
```

String methods

```python
ss = "Hello, World"
print(ss.upper())

tt = ss.lower()
print(tt)
```

More Methods: https://shorturl.at/flST1

Count and replace, replace is very important!

```python
ss = "    Hello, World    "

els = ss.count("l")
print(els)
news = ss.replace("o", "***")
print(news)
```

String format method – Fill in the blanks like

- Hello ____________

```
person = input('Your name: ')
greeting = 'Hello {}!'.format(person)
print(greeting)
```

Example with float, type-casting and format

```python
origPrice = float(input('Enter the original price: $'))
discount = float(input('Enter discount percentage: '))
newPrice = (1 - discount/100)*origPrice
calculation = '${} discounted by {}% is ${}.'.format(origPrice, discount, newPrice)
print(calculation)
```

String length, there is common error, index out of range, let's try

```
fruit = "Banana"
print(len(fruit))
```

```
fruit = "Khokhri"
sz = len(fruit)
lastch = fruit[sz-1]
print(lastch)
```

The slice operator

- A substring of a string is called a slice. Selecting a slice is similar to selecting a character:

```
singers = "Peter, Paul, and Mary"
print(singers[0:5])
print(singers[7:11])
print(singers[17:21])
```

The slice operator [n:m] returns the part of the string from the n'th character to the m'th character, including the first but excluding the last.

String comparison

- The comparison operators also work on strings. To see if two strings are equal you simply write a boolean expression using the equality operator.

```
word = "banana"
if word == "banana":
    print("Yes, we have bananas!")
else:
    print("Yes, we have NO bananas!")
```

```python
word = "zebra"

if word < "banana":
    print("Your word, " + word + ", comes before banana.")
elif word > "banana":
    print("Your word, " + word + ", comes after banana.")
else:
    print("Yes, we have no bananas!")
```

Apple != Apple

```
print("apple" < "banana")

print("apple" == "Apple")
print("apple" < "Apple")
```

String are immutable

One final thing that makes strings different from some other Python collection types is that you are not allowed to modify the individual characters in the collection. It is tempting to use the [] operator on the left side of an assignment, with the intention of changing a character in a string. For example, in the following code, we would like to change the first letter of greeting.

```
greeting = "Hello, world!"
greeting[0] = 'J'            # ERROR!
print(greeting)
```

Then what is solution to this problem?

- Strings are immutable, which means you cannot change an existing string. The best you can do is create a new string that is a variation on the original.

```
greeting = "Hello, world!"
newGreeting = 'J' + greeting[1:]
print(newGreeting)
print(greeting)
```

Traversal and the for Loop: By Item

- Often we start at the beginning, select each character in turn, do something to it, and continue until the end. This pattern of processing is called a **traversal**.

```
for aname in ["Joe", "Amy", "Brad", "Angelina", "Zuki", "Thandi", "Paris"]:
    invitation = "Hi " + aname + ".  Please come to my party on Saturday!"
    print(invitation)
```

Traversal and the for Loop: By Index

```python
fruit = "apple"
for idx in range(len(fruit)):
    print(fruit[idx])
```

Traversal and the while Loop

```
fruit = "apple"

position = 0
while position < len(fruit):
    print(fruit[position])
    position = position + 1
```

The in and not in operators

- The in operator tests if one string is a substring of another:

```
print('p' in 'apple')
print('i' in 'apple')
print('ap' in 'apple')
print('pa' in 'apple')
```

```
print('x' not in 'apple')
```

The Accumulator Pattern with Strings

```
def removeVowels(s):
    vowels = "aeiouAEIOU"
    sWithoutVowels = ""
    for eachChar in s:
        if eachChar not in vowels:
            sWithoutVowels = sWithoutVowels + eachChar
    return sWithoutVowels

print(removeVowels("compsci"))
print(removeVowels("aAbEefIijOopUus"))
```

Dry run: https://shorturl.at/gkl79

Looping and counting

```
def count(text, aChar):
    lettercount = 0
    for c in text:
        if c == aChar:
            lettercount = lettercount + 1
    return lettercount

print(count("banana","a"))
```

Its your turn now!

""" Find and return the index of achar in astring. Return -1 if achar does not occur in astring. """

Solution :)

```python
def find(astring, achar):
    ix = 0
    found = False
    while ix < len(astring) and not found:
        if astring[ix] == achar:
            found = True
        else:
            ix = ix + 1
    if found:
        return ix
    else:
        return -1
print(find("Compsci", "p"))
```

Slides & Material

razaulmustafa.us/cs148/