PYTHON

Lecture - 08

Lecture Topics

- String

Python - String Manipulation (RECAP)

Declaration of a string

- **Single Quotes**
  >>> spam = 'Say hi to Bob\'s mother.'

- **Double Quotes**
  >>> spam = "That is Alice's cat."

- **Triple Quotes**
  print('''Dear Alice, Eve's cat has been arrested for catnapping, cat burglary, and extortion. Sincerely,
  Bob''')

Python - String Manipulation (RECAP)

Escape Characters in string

Escape character	Prints as
\'	Single quote
\"	Double quote
\t	Tab
\n	Newline (line break)
\\	Backslash

```
>>> print("Hello there!\nHow are you?\nI\'m doing fine.")
Hello there!
How are you?
I'm doing fine.
```

Python - String Manipulation (RECAP)

Raw Strings

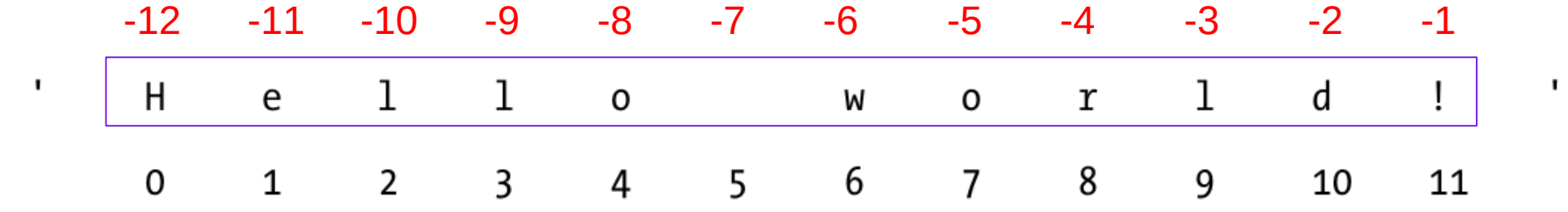
Place an **r** before the beginning quotation mark of a string to make it a raw string.
A **raw string** completely ignores all escape characters and prints any backslash that appears in the string.

```
>>> print(r'That is Carol\'s cat.')
That is Carol\'s cat.
```

Python - String Manipulation

Indexing and Slicing Strings

Think of the string **'Hello world!'** as a list and each character in the string as an item with a corresponding index.




```
>>> spam = 'Hello world!'
>>> spam[0]
'H'
>>> spam[4]
'o'
>>> spam[-1]
'!'
```

```
>>> spam[0:5]
'Hello'
>>> spam[:5]
'Hello'
>>> spam[6:]
'world!'
```

Python - String Manipulation (Theory Recap)

Indexing and Slicing Strings

str = “ a b c d e f ”
Index→ 0 1 2 3 4 5

str[start : end] → return substring of **str** from index **start** to **end-1.**

str[: idx] → return substring of **str** from **beginning** to index **idx-1.**

str[idx :] → return substring of **str** from index **idx** to **end** of the string**.**

Python - String Manipulation (Theory Recap)

Indexing and Slicing Strings

str = “ a b c d e f”
Index→ 0 1 2 3 4 5

```
print( str[2:4] )        # ‘cd’

print( str[1:] )         # ‘abcdef’

print( str[:3] )         # ‘abc’
```

Python - String Manipulation

The in and not in Operators with Strings

The **in** and **not in** operators can be used with strings just like with list values. An expression with two strings joined using **in** or **not in** will evaluate to a Boolean **True** or **False**.

```
>>> 'Hello' in 'Hello World'
True
>>> 'Hello' in 'Hello'
True
>>> 'HELLO' in 'Hello World'
False
>>> '' in 'spam'
True
>>> 'cats' not in 'cats and dogs'
False
```

Python - String Manipulation

The upper(), lower() String Methods

The **upper()** and **lower()** string methods return a new string where all the letters in the original string have been converted to **uppercase** or **lower-case**, respectively.
Non-letter characters in the string remain **unchanged**.

```
>>> spam = 'Hello world!'
>>> spam = spam.upper()
>>> spam
'HELLO WORLD!'
>>> spam = spam.lower()
>>> spam
'hello world!'
```

Python - String Manipulation

The isupper(), islower() String Methods

The **isupper()** and **islower()** methods will return a Boolean **True** value if the string has **at least one letter** and all the letters are **uppercase** or **lowercase**, respectively. Otherwise, the method returns **False**.

```
>>> spam = 'Hello world!'
>>> spam.islower()
False
>>> spam.isupper()
False
>>> 'HELLO'.isupper()
True
>>> 'abc12345'.islower()
True
>>> '12345'.islower()
False
>>> '12345'.isupper()
False
```

Python - String Manipulation

Exercise – 01

```
>>> 'Hello'.upper()
'HELLO'
>>> 'Hello'.upper().lower()
'hello'
>>> 'Hello'.upper().lower().upper()
'HELLO'
>>> 'HELLO'.lower()
'hello'
>>> 'HELLO'.lower().islower()
True
```

Python - String Manipulation

Exercise – 02

```
print('How are you?')
feeling = input()
if feeling.lower() == 'great':
    print('I feel great too.')
else:
    print('I hope the rest of your day is good.')
```

```
How are you?
GREat
I feel great too.
```

Python - String Manipulation

The isX String Methods

- **isalpha()** → returns True if the string consists only of letters and is not blank.
- **isalnum()** → returns True if the string consists only of letters and numbers and is not blank.
- **isdecimal()** → returns True if the string consists only of numeric characters and is not blank.
- **isspace()** → returns True if the string consists only of spaces, tabs, and new- lines and is not blank.
- **istitle()** → returns True if the string consists only of words that begin with an uppercase letter followed by only lowercase letters.

Python - String Manipulation

The isX String Methods

```
>>> 'hello'.isalpha()
True
>>> 'hello123'.isalpha()
False
>>> 'hello123'.isalnum()
True
>>> 'hello'.isalnum()
True
>>> '123'.isdecimal()
True
>>> '    '.isspace()
True
>>> 'This Is Title Case'.istitle()
True
>>> 'This Is Title Case 123'.istitle()
True
>>> 'This Is not Title Case'.istitle()
False
>>> 'This Is NOT Title Case Either'.istitle()
False
```

Python - String Manipulation

The startswith() and endswith() String Methods

The **startswith()** and **endswith()** methods return **True** if the string value they are called on **begins** or **ends** (respectively) with the string passed to the method; otherwise, they return **False**.

```
>>> 'Hello world!'.startswith('Hello')
True
>>> 'Hello world!'.endswith('world!')
True
>>> 'abc123'.startswith('abcdef')
False
>>> 'abc123'.endswith('12')
False
>>> 'Hello world!'.startswith('Hello world!')
True
>>> 'Hello world!'.endswith('Hello world!')
True
```

Python - String Manipulation

The join() String Methods

The join() method is useful when you have a list of strings that need to be joined together into a single string value.

The join() method is called on a string, gets passed a list of strings, and returns a string.

The returned string is the concatenation of each string in the passed-in list.

```
>>> ', '.join(['cats', 'rats', 'bats'])
'cats, rats, bats'
>>> ' '.join(['My', 'name', 'is', 'Simon'])
'My name is Simon'
>>> 'ABC'.join(['My', 'name', 'is', 'Simon'])
'MyABCnameABCisABCSimon'
```

Python - String Manipulation

The split() String Methods

The split() method does the opposite.

It’s called on a string value and returns a list of strings.

```
>>> 'My name is Simon'.split()
['My', 'name', 'is', 'Simon']
```

By default, the string 'My name is Simon' is split wherever **whitespace-characters** such as the **space**, **tab**, or **newline** characters are found.

Python - String Manipulation

The split() String Methods

You can pass a **delimiter string** to the split() method to specify a different string to split upon.

```
>>> 'MyABCnameABCisABCSimon'.split('ABC')
['My', 'name', 'is', 'Simon']
>>> 'My name is Simon'.split('m')
['My na', 'e is Si', 'on']
```

Python - String Manipulation

Justifying Text with rjust(), ljust(), and center()

The rjust() and ljust() string methods return a padded version of the string they are called on, with spaces inserted to justify the text.

The first argument to both methods is an integer length for the justified string.

```
>>> 'Hello'.rjust(10)
'     Hello'
>>> 'Hello'.rjust(20)
'               Hello'
>>> 'Hello World'.rjust(20)
'         Hello World'
>>> 'Hello'.ljust(10)
'Hello     '
```

Python - String Manipulation

Justifying Text with rjust(), ljust(), and center()

An optional second argument to rjust() and ljust() will specify a fill character other than a space character.

```
>>> 'Hello'.rjust(20, '*')
'***************Hello'
>>> 'Hello'.ljust(20, '-')
'Hello---------------'
```

Python - String Manipulation

Justifying Text with rjust(), ljust(), and center()

The center() string method works like ljust() and rjust() but centers the text rather than justifying it to the left or right.

```
>>> 'Hello'.center(20)
'       Hello        '
>>> 'Hello'.center(20, '=')
'=======Hello========'
```

Python - String Manipulation

Removing Whitespace with strip(), rstrip(), and lstrip()

The strip() string method will return a new string without any whitespace characters at the beginning or end.

The lstrip() and rstrip() methods will remove whitespace characters from the left and right ends, respectively.

```
>>> spam = '    Hello World    '
>>> spam.strip()
'Hello World'
>>> spam.lstrip()
'Hello World    '
>>> spam.rstrip()
'    Hello World'
```

Python - String Manipulation

Removing Whitespace with strip(), rstrip(), and lstrip()

Optionally, a string argument will specify which characters on the ends should be stripped.

```
>>> spam = 'SpamSpamBaconSpamEggsSpamSpam'
>>> spam.strip('ampS')
'BaconSpamEggs'
```

Before and after *BaconSpamEggs* all characters within *ampS*

Exercise Time

Exercise – 7.1

Create a function that takes **a list** of int numbers as a parameter and returns its **sum**.

Exercise – 7.1 (ans)

```
def findSum(x):
    sum = 0
    for i in x:
        sum+=i
    return sum


List = [2,3,7,10]
print(findSum(List))
```

Exercise – 7.2

Create a function that takes **two** of **int** numbers as a parameter and returns thair **GCD**.

Exercise – 7.2 (ans)

```
def GCD(x,y):
    n = min(x,y)
    gcd = 0
    for i in range(1,n+1):
        if x%i == 0 and y%i == 0:
            gcd = i
    return gcd


ans = GCD(8,12)
print(ans)
```

Exercise – 7.3

Build a **module** called **myString.py** that contain the following functions (each of them takes a string as parameter) —
vowelCount() → returns the count of vowel in the string.
uniqueChar() → returns the count of unique char of the string.

And then import and call the functions from another python file.

For Example:
vowelCount(“school”) → return 2, as there are 2 vowels.
uniqueChar(“exercise”) → return 6, as there are 6 unique char.

Exercise – 7.3 (ans)

```
#save file as myString.py
def vowelCount(str):
    cnt = 0
    for ch in str:
        if(ch=='a' or ch=='e' or ch=='i' or ch=='o' or ch=='u'):
            cnt+=1
    return cnt

def uniqueChar(str):
    List = []
    for i in range(26):
        List.append(0)
    for ch in str:
        idx = int(ord(ch)) - int(ord('a'))
        List[idx]=1
    cnt = 0
    for i in List:
        if(i>0):
            cnt+=1
    return cnt
```

Exercise – 7.3 (ans cont.)

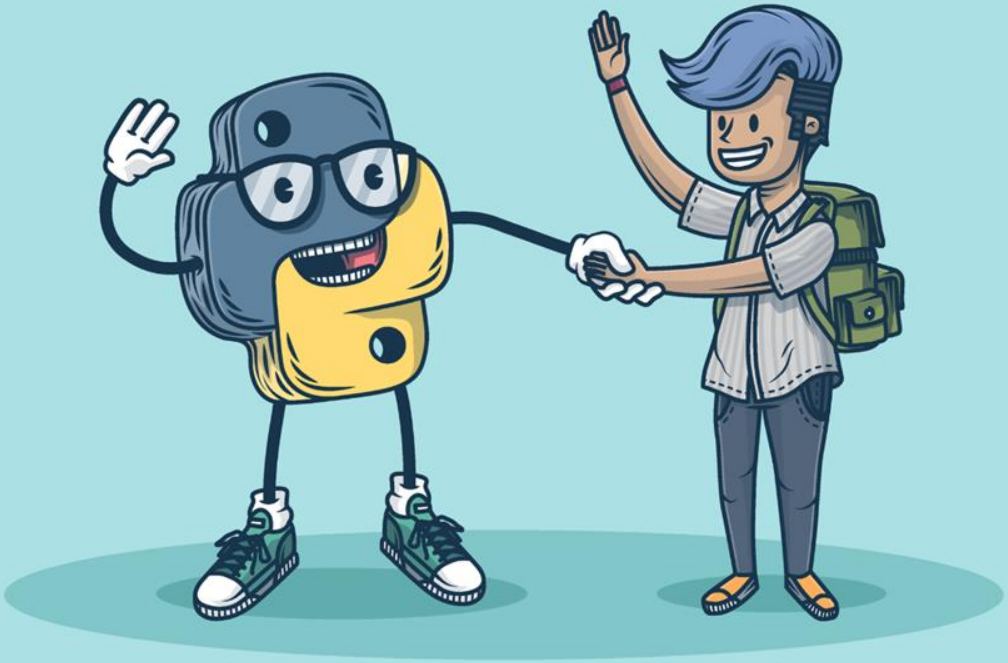
```
#In another python file, write the code and run

import myString

print(myString.vowelCount("school"))
print(myString.uniqueChar("exercise"))
```

Resources

- https://www.tutorialspoint.com/python/index.htm
- https://www.w3resource.com/python/python-tutorial.php
- https://www.w3resource.com/python-exercises/string/
- https://www.w3schools.com/python/
- https://www.geeksforgeeks.org/python-programming-language/
- https://youtu.be/t2_Q2BRzeEE?si=OO6J_YNCZykedqsT
- https://realpython.com/
- Head First Python, 3rd Edition by Paul Barry
- Automate the Boring Stuff with Python By Al Sweigart.



Thank You