

Strings in Python

PairProgramming Exercise, DSE5002

Beija Richardson March 30 2025,

String variables in Python are arrays, we can access the letters in them by index

We will import the re package, that allows us to carry out string manipulation using regular expression (regex), that we will see more about later

Helpful source

https://www.w3schools.com/python/python_ref_string.asp

```
In [6]: import re
```

```
In [7]: my_string="String variables in Python are used to store sequences of characters."
```

```
In [8]: # accessing a region of the string
```

```
my_string[7:15]
```

```
Out[8]: 'variable'
```

Question/Action

access the word store in my_string

```
In [10]: list(my_string)
```

```
Out[10]: ['S',  
          't',  
          'r',  
          'i',  
          'n',  
          'g',  
          ' ',  
          'v',  
          'a',  
          'r',  
          'i',  
          'a',  
          'b',  
          'l',  
          'e',  
          's',  
          ' ',  
          'i',  
          'n',  
          ' ',  
          'p',  
          'y',  
          't',  
          'h',  
          'o',  
          'n',  
          ' ',  
          'a',  
          'r',  
          'e',  
          ' ',  
          'u',  
          's',  
          'e',  
          'd',  
          ' ',  
          't',  
          'o',  
          ' ',  
          's',  
          't',  
          'o',  
          'r',  
          'e',  
          ' ',  
          's',  
          'e',  
          'q',  
          'u',  
          'e',  
          'n',  
          'c',  
          'e',  
          's',  
          ' ',  
          'o',
```

```
'f',  
'i',  
'c',  
'h',  
'a',  
'r',  
'a',  
'c',  
't',  
'e',  
'r',  
's',  
'.']
```

comparing strings

can be done with standard operations

```
In [ ]: "alpha"=="beta"
```

```
In [ ]: "alpha"<"beta"
```

```
In [ ]: #string methods
```

These are a number of standardized ways of modifying strings

```
In [ ]: myword="Python rocks!"  
  
print(myword.lower())  
print(myword.upper())  
print(myword.title())
```

find, replace, strip, split and index

These are basic operations on strings. The built-in string functions in Python are very limited, they act like the find/replace commands in Word or Excel, rather than using the more sophisticated regular expression patterns we will be discussing shortly

```
In [ ]: # find returns the index location where the pattern string is found  
  
myword.find("rock")
```

```
In [ ]: # find will return a -1 if the word is not in the string  
  
myword.find("chill")
```

```
In [ ]: #replace replaces a target word with another word
```

```
In [ ]: myword.replace('Python', "C")
```

```
In [ ]: myword.replace('Rust','C')
```

Strip

Strip removes excess leading or trailing white space

Handy for cleaning up a string

```
In [ ]: yourword=" Once upon a regex manual "  
yourword.strip()
```

```
In [ ]: #split  
  
This splits a string into pieces based on a delimited, the delimiter is removed
```

```
In [ ]: theirstring="Alpha, Beta, Gamma"  
theirstring.split(",")
```

```
In [ ]: theirstring.split(" ")
```

```
In [ ]: #index, it's not clear to me how index differs from find  
myword.index("roc")
```

Regular Expressions, or Regex

In regex, we create a search pattern that is used in

-find

-replace

-string splitting

```
In [ ]: import re  
  
#re is a python package for searching with regex  
my_string="Hey, where are the anchovies?"  
pattern="are"  
re.search(pattern,my_string)
```

This means that the pattern "are" can be found from the location 27 to 30 in my_string

If the pattern is not in the string, we get a -1

```
In [ ]: pattern="arc"  
  
re.search(pattern,my_string)
```

Wildcards

.- this is a wildcard for a single character

- -this is a wildcard for any number of characters

```
In [ ]: pattern="anc.ovies"  
  
res=re.search(pattern,my_string)  
  
print(res)  
  
print(res.span())
```

Splitting

```
In [ ]: pattern=" "  
re.split(pattern,my_string)
```

```
In [ ]: # replacement  
  
pattern="Hey"  
newpattern="Wow"  
  
re.sub(pattern, newpattern, my_string)
```

```
In [ ]:
```