Strings in Python

PairProgramming Exercise, DSE5002

Feb 2025,

String variables in Pythons 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 [3]:
```
import re
```

In [4]:
```
my_string="String variables in Python are used to store sequences of characters."
```

In [5]:
```
# accessing a region of the string

my_string[7:15]
```

Out[5]:
```
'variable'
```

Question/Action

access the word store in my_string

In [8]:
```
my_string[39:45]
```

Out[8]:
```
'store '
```

In [ ]:
```
## comparing strings

can be done with standard operations
```

In [9]:
```
"alpha"=="beta"
```

Out[9]:
```
False
```

In [10]:
```
"alpha"<"beta"
```

Out[10]:
```
True
```

In [12]:
```
#string methods

These are a number of standardized ways of modifying strings
```

```
  Cell In[12], line 3
    These are a number of standardized ways of modifying strings
          ^
SyntaxError: invalid syntax
```

In [11]:
```
myword="Python rocks!"

print(myword.lower())
print(myword.upper())
print(myword.title())
```

```
python rocks!
PYTHON ROCKS!
Python Rocks!
```

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/relace commands in Word or Excel, rather than using the more sophisticated regular expression patterns we will be discussing shortly

In [13]:
```
# find returns the index location where the pattern string is found

myword.find("rock")
```

Out[13]: 7

In [14]:
```
# find will return a -1 if the word is not in the string

myword.find("chill")
```

Out[14]: -1

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

In [15]:
```
myword.replace('Python',"C")
```

Out[15]: 'C rocks!'

In [16]:
```
myword.replace('Rust',"C")
```

Out[16]: 'Python rocks!'

Strip

Strip removes excess leading or trailing white space

Handy for cleaning up a string

In [17]:
```
yourword=" Once upon a regex manual "

yourword.strip()
```

Out[17]:
```
'Once upon a regex manual'
```

In [18]:
```
#split

This splits a string into pieces based on a delimited, the delimiter is removed
```

```
  Cell In[18], line 3
    This splits a string into pieces based on a delimited, the delimiter is removed
         ^
SyntaxError: invalid syntax
```

In [19]:
```
theirstring="Alpha, Beta, Gamma"

theirstring.split(",")
```

Out[19]:
```
['Alpha', ' Beta', ' Gamma']
```

In [20]:
```
theirstring.split(" ")
```

Out[20]:
```
['Alpha,', 'Beta,', 'Gamma']
```

In [21]:
```
#index,  it's not clear to me how index differs from find

myword.index("roc")
```

Out[21]:
```
7
```

Regular Expressions, or Regex

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

-find

-replace

-string splitting

In [22]:
```
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)
```

Out[22]: <re.Match object; span=(11, 14), match='are'>

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 [23]:
```
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 [24]:
```
pattern="anc.ovies"

res=re.search(pattern,my_string)

print(res)

print(res.span())
```

```
<re.Match object; span=(19, 28), match='anchovies'>
(19, 28)
```

Splitting

In [25]:
```
pattern=" "
re.split(pattern,my_string)
```

Out[25]: ['Hey,', 'where', 'are', 'the', 'anchovies?']

In [26]:
```
# replacement

pattern="Hey"
newpattern="Wow"

re.sub(pattern, newpattern, my_string)
```

Out[26]: 'Wow, where are the anchovies?'

In [ ]: