Strings data type in Python

Data type in Python – **Numbers**, **Strings, Lists, Tuple, Dictionary**

Numbers, Strings, Tuple – Immutable data type

Lists, Dictionary – Mutable data type

STRING

- Strings are a contiguous set of characters enclosed within quotes.
- The quotes can be single, double, or triple.

Escape Sequence: Non- printable character having some meaning to the python interpreter.

\n	**newline**
\t	**tab**
\”	**double quotes**
\’	**single quote**

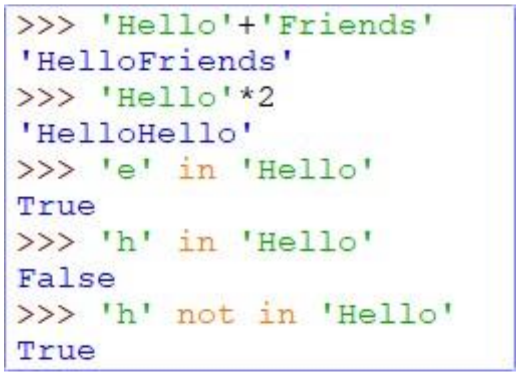
What are the different ways to assign the message – **Hello, “How are you” ?** in a variable str?

str = ‘**Hello, “How are you” ?’**
str = “Hello, \“How are you\” ?”
str = ‘‘‘**Hello, “How are you”**
?’’’
str = “““**Hello, “How are you” ?”””**

String operations

Expression	**Results**	**Description**
‘Hello!’ + ‘Friends’	HelloFriends	**Concatenation** – joins the strings into a single string
‘Hello’ * 2	HelloHello	**Repetition** – repeats the string

‘e’ in ‘Hello’ ‘h’ in ‘Hello’ ‘h’ not in ‘Hello’	True False True	**Membership** – returns True if the character is a member of string otherwise False.



```
>>> 'Hello'+'Friends'
'HelloFriends'
>>> 'Hello'*2
'HelloHello'
>>> 'e' in 'Hello'
True
>>> 'h' in 'Hello'
False
>>> 'h' not in 'Hello'
True
```

Indexing in strings

Each character in a string is automatically assigned an index number. Two sets of indices are used in Python.

• Positive integers are used to index from left to right.

• Negative integers are used to index from right to left.

Character	**H**	**A**	**P**	**P**	**I**	**N**	**E**	**S**	**S**
Index from left to right	0	1	2	3	4	5	6	7	8
Index from right to left	-9	-8	-7	-6	-5	-4	-3	-2	-1

String slicing

Slicing is an act of extracting a piece of a string from the original string. The general format is:

str [startIndex : pastIndex]

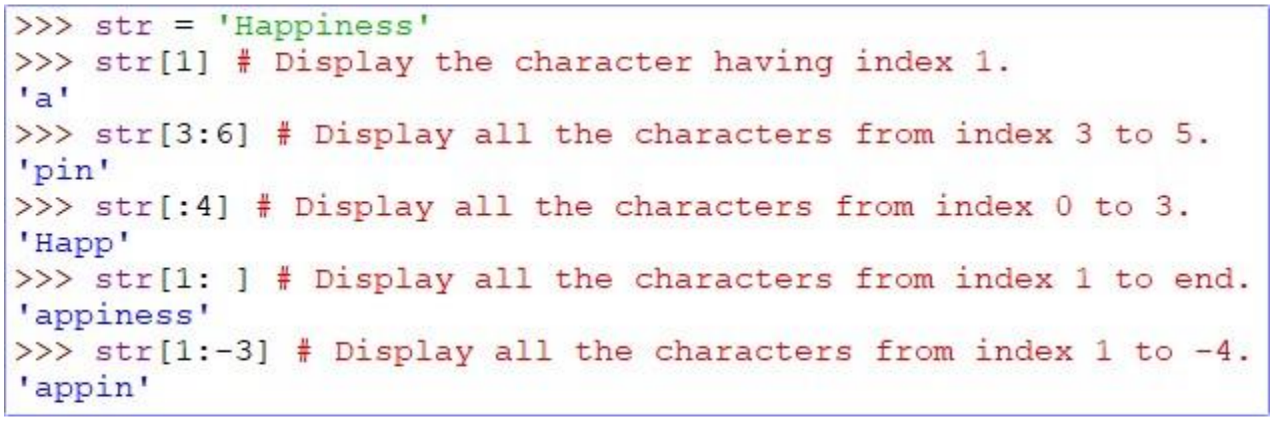
- ➢ This refers to the substring of str starting at startIndex and stop before pastIndex.
- ➢ If you omit the startIndex, substring starts from the beginning (0th location).

- If you omit the pastIndex, substring starts from the startIndex and goes to the end.

Slicing operation on Strings

Slicing operation	**Explanation**
str[1]	It returns the second character of the string str.
str[3:6]	It returns the substring by accessing the character from the third index to fifth index, i.e., before the sixth index.
str[:4]	Omitting the first index means that it starts accessing characters from the beginning, i.e., 0th index and stops at index 3.
str[1:]	Omitting the second index means that it starts accessing characters from the first index, i.e., 1 and stops at the end.
str[1: -3]	It starts accessing characters at the first index and stops before the second index, i.e., at -4, the fourth character from the right.

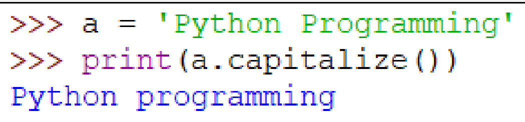
Figure 6.2 shows output of slicing operation on string.

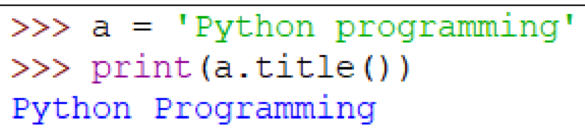


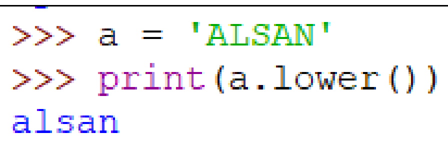
```
>>> str = 'Happiness'
>>> str[1] # Display the character having index 1.
'a'
>>> str[3:6] # Display all the characters from index 3 to 5.
'pin'
>>> str[:4] # Display all the characters from index 0 to 3.
'Happ'
>>> str[1: ] # Display all the characters from index 1 to end.
'appiness'
>>> str[1:-3] # Display all the characters from index 1 to -4.
'appin'
```

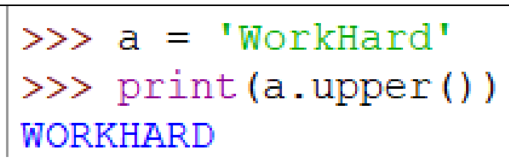
String functions in Python

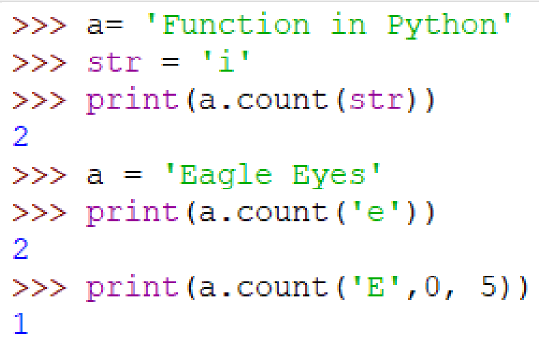
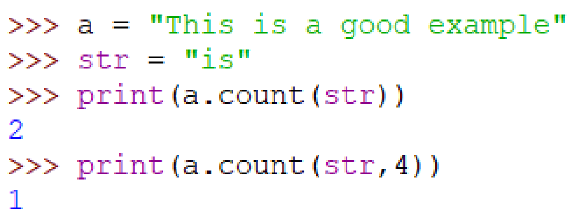
Function syntax and details	Function description
Conversion Functions	

capitalize() – it capitalize the very first letter of the string and rest of the characters are in lower case.	 `>>> a = 'Python Programming'`
	`>>> print(a.capitalize())`
	`Python programming`

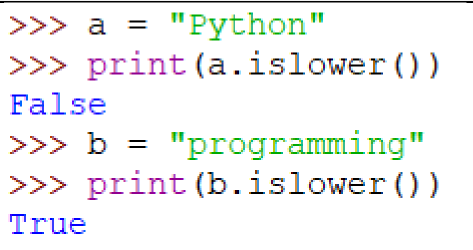
title() – It returns all the words start with uppercase.	 `>>> a = 'Python programming'`
	`>>> print(a.title())`
	`Python Programming`

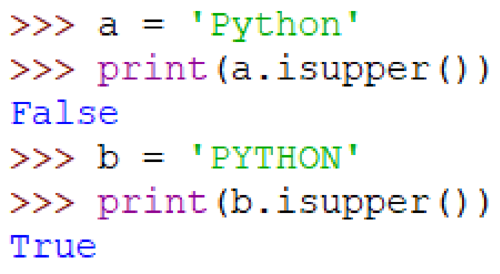
lower() – It converts all the characters of the string to lowercase.	 `>>> a = 'ALSAN'`
	`>>> print(a.lower())`
	`alsan`

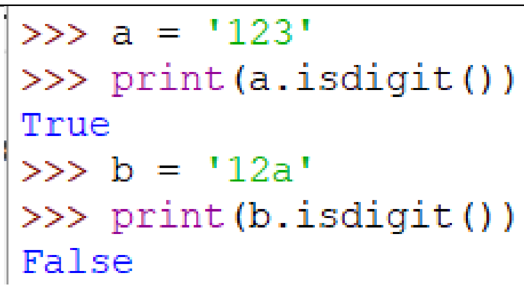
upper() – it converts all the characters of the string to uppercase.	 `>>> a = 'WorkHard'`
	`>>> print(a.upper())`
	`WORKHARD`

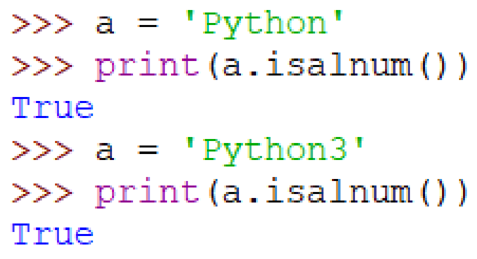
count(str [, beg [, end]]) – It returns the number of times substring 'str' occurs in the range [beg, end] if beg and end index are given else the search continues in full String.
 	
Note: Search is case-sensitive.
	 `>>> a= 'Function in Python'`
	`>>> str = 'i'`
	`>>> print(a.count(str))`
	`2`
	`>>> a = 'Eagle Eyes'`
	`>>> print(a.count('e'))`
	`2`
	`>>> print(a.count('E',0, 5))`
	`1`
	 `>>> a = "This is a good example"`
	`>>> str = "is"`
	`>>> print(a.count(str))`
	`2`
	`>>> print(a.count(str,4))`
	`1`

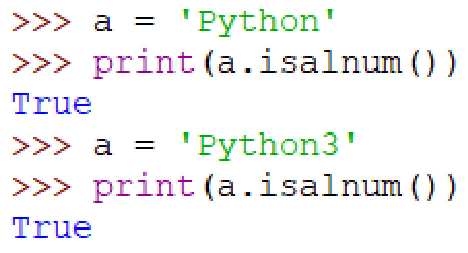
Comparison Functions	

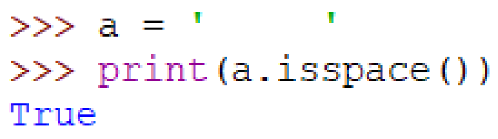
islower() – It returns ‘True’ if all the characters in the String are in lowercase. If any of the char is in uppercase, it will return False.	 `>>> a = "Python"`
	`>>> print(a.islower())`
	`False`
	`>>> b = "programming"`
	`>>> print(b.islower())`
	`True`

isupper() – It returns ‘True’ if all the characters in the String are in uppercase. If any of the char is in lowercase, it will return False.	 `>>> a = 'Python'`
	`>>> print(a.isupper())`
	`False`
	`>>> b = 'PYTHON'`
	`>>> print(b.isupper())`
	`True`

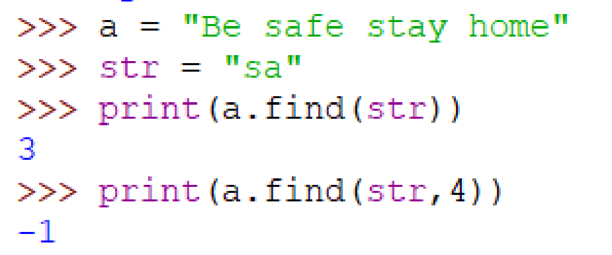
isdigit() - It returns TRUE if the provided string contains only digits and returns FALSE otherwise.	 `>>> a = '123'`
	`>>> print(a.isdigit())`
	`True`
	`>>> b = '12a'`
	`>>> print(b.isdigit())`
	`False`

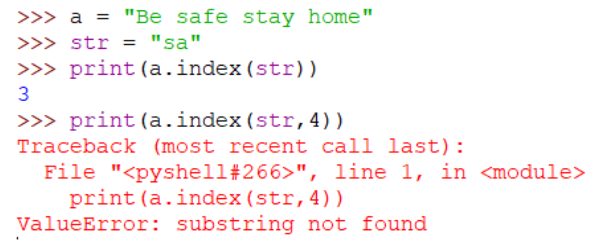
isalpha() – It returns true if a string contains at least one alphabetic character, otherwise it returns false.	 `>>> a = 'Python'`
	`>>> print(a.isalnum())`
	`True`
	`>>> a = 'Python3'`
	`>>> print(a.isalnum())`
	`True`

isalnum() – It returns true if string contains at least one character and all the other characters are either alphabetic or decimal digits, otherwise it returns False.	 `>>> a = 'Python'`
	`>>> print(a.isalnum())`
	`True`
	`>>> a = 'Python3'`
	`>>> print(a.isalnum())`
	`True`

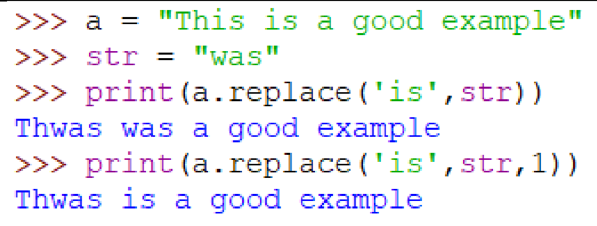
isspace() – it returns TRUE if the string provided has only space characters, otherwise it returns FALSE.	 `>>> a = '      '`
	`>>> print(a.isspace())`
	`True`

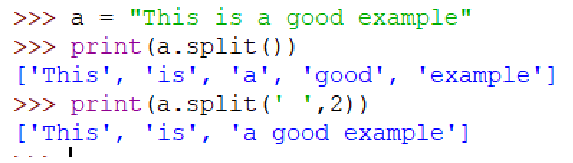
Search Functions	

find(str [, i [, j]]) – It searches for ‘str’ in complete String (if i and j not defined) or in a sub-string of String (if i and j are defined).This function returns the index if ‘str’ is found else returns ‘-1’.
 	
Here, i=search starts from this index, j=search ends at this index.
	
	`>>> a = "Be safe stay home"`
	`>>> str = "sa"`
	`>>> print(a.find(str))`
	`3`
	`>>> print(a.find(str,4))`
	`-1`

index(str [, i [, j]]) – It is the same as find() but raises the ‘ValueError’ if str does not exist.	
	`>>> a = "Be safe stay home"`
	`>>> str = "sa"`
	`>>> print(a.index(str))`
	`3`
	`>>> print(a.index(str,4))`
	`Traceback (most recent call last):`
	`  File "<pyshell#266>", line 1, in <module>`
	`    print(a.index(str,4))`
	`ValueError: substring not found`

String Substitution functions	

replace(old, new [, count]) – It replaces all the occurrences of substring ‘old’ with ‘new’ in the String.
 	
If the count is available, then only ‘count’ number of occurrences of ‘old’ will be replaced with the ‘new’ var.
 	
Where old =substring to replace,
new =substring
	
	`>>> a = "This is a good example"`
	`>>> str = "was"`
	`>>> print(a.replace('is',str))`
	`Thwas was a good example`
	`>>> print(a.replace('is',str,1))`
	`Thwas is a good example`

split([sep[,maxsplit]]) – It returns a list of substring obtained after splitting the String with ‘sep’ as a delimiter.
Where, sep= delimiter, the default is space, maxsplit= number of splits to be done
	
	`>>> a = "This is a good example"`
	`>>> print(a.split())`
	`['This', 'is', 'a', 'good', 'example']`
	`>>> print(a.split(' ',2))`
	`['This', 'is', 'a good example']`

Miscellaneous string functions	

len(string) – It returns the length of a string.	`>>> a = "     assaxaszylt        "`
	`>>> l = len(a)`
	`>>> print(l)`
	`24`

lstrip([chars]) – it returns a string after removing the characters from the beginning of the String.
Where chars = character to be trimmed from the string.
The default is the whitespace character.
	`>>> a = "     assaxaszylt"`
	`>>> print(a.lstrip())`
	`assaxaszylt`

rstrip([chars]) – it returns a string after removing the characters from the end of the String.	`>>> a = "assaxaszylt           "`
	`>>> print(a.rstrip())`
	`assaxaszylt`

strip([chars]) – It returns a string after removing the characters from the beginning and end of the String.	`>>> a = "     assaxaszylt        "`
	`>>> print(a.strip())`
	`assaxaszylt`

The partition() method searches for the first occurrence of the specified string, and splits the string into a tuple containing three elements.
 	
➢ The first element contains the part before the specified string.
➢ The second element contains the specified string.
➢ The third element contains the part after the string.
	`>>> str = "Stay home stay safe"`
	`>>> str.partition("stay")`
	`('Stay home ', 'stay', ' safe')`
	`>>> str.partition("keep")`
	`('Stay home stay safe', '', '')`
	`>>> str.partition("safe")`
	`('Stay home stay ', 'safe', '')`
	`>>> str.partition("Stay")`
	`('', 'Stay', ' home stay safe')`