LISTS

Agenda

- What is list and Getting list
- List are mutable
- Traversing a list
- List operations
- List slices
- List methods
- List lambda,map filter and reduce
- Deleting elements
- Lists and Strings

What is list and Getting list

- **List:**List contain items of different data types. Lists are mutable i.e., modifiable.
- The values stored in List are separated by commas(,) and enclosed within a square brackets([]). We can store different type of data in a List.
- Value stored in a List can be retrieved using the slice operator([] and [:]).
- The plus sign (+) is the list concatenation and asterisk(*) is the repetition operator.

How to create a list

- In Python programming, a list is created by placing all the items (elements) inside a square bracket [ ], separated by commas.
- It can have any number of items and they may be of different types (integer, float, string etc.).
- Example1: Empty List

Non-Empty List

```
Python 3.6.1 Shell
File  Edit  Shell  Debug  Options  Window  Help
Python 3.6.1 (v3.6.1:69c0db5, Mar 21 2017, 17:54:52) [MSC v.1900 32 bit (Intel)]
 on win32
Type "copyright", "credits" or "license()" for more information.
>>> names = ["C", "Python", "Android", "Java"]
>>> names
['C', 'Python', 'Android', 'Java']
>>> |
Ln: 6  Col: 4
```

The above example is a non empty list because we did passed some values in list.

a **term** called index in a python for list which indicates the position of the value in the list. List having two types of index are there i.e.,

1. Forward index: When index starts from 0 then values displaying from left to right.

Ex: 0, 1, 2, 3, 4…

2. Backward index: When index starts from -1 then values displaying from right to left.

Ex: -1, -2, -3, -4, ……..

Continue…….

Python 3.4.4 Shell

File Edit Shell Debug Options Window Help

```
Python 3.4.4 (v3.4.4:737efcadf5a6, Dec 20 2015, 20:20:57) [MSC v.1600 64 bit (AM
D64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> list=['aman',678,20.4,'saurav']
>>> list1=[456,'rahul']
>>> list
['aman', 678, 20.4, 'saurav']
>>> list[1:3]
[678, 20.4]
>>> list1+list
[456, 'rahul', 'aman', 678, 20.4, 'saurav']
>>> list*2
['aman', 678, 20.4, 'saurav', 'aman', 678, 20.4, 'saurav']
>>>
```

Ln: 13 Col: 4

Forward and backward index

Python 3.6.1 Shell

File Edit Shell Debug Options Window Help

```
Python 3.6.1 (v3.6.1:69c0db5, Mar 21 2017, 17:54:52) [MSC v.1900 32 bit (Intel)]
 on win32
Type "copyright", "credits" or "license()" for more information.
>>> names = ["C", "Python", "Android", "Java"]
>>> names
['C', 'Python', 'Android', 'Java']
>>> names[0]
'C'
>>> names[2]
'Android'
>>> |
```

Ln: 10 Col: 4

Python 3.6.1 Shell

File Edit Shell Debug Options Window Help

```
Python 3.6.1 (v3.6.1:69c0db5, Mar 21 2017, 17:54:52) [MSC v.1900 32 bit (Intel)]
 on win32
Type "copyright", "credits" or "license()" for more information.
>>> names = ["C","Python", "Android", "Java"]
>>> names
['C', 'Python', 'Android', 'Java']
>>> names[-1]
'Java'
>>> names[-4]
'C'
>>> |
```

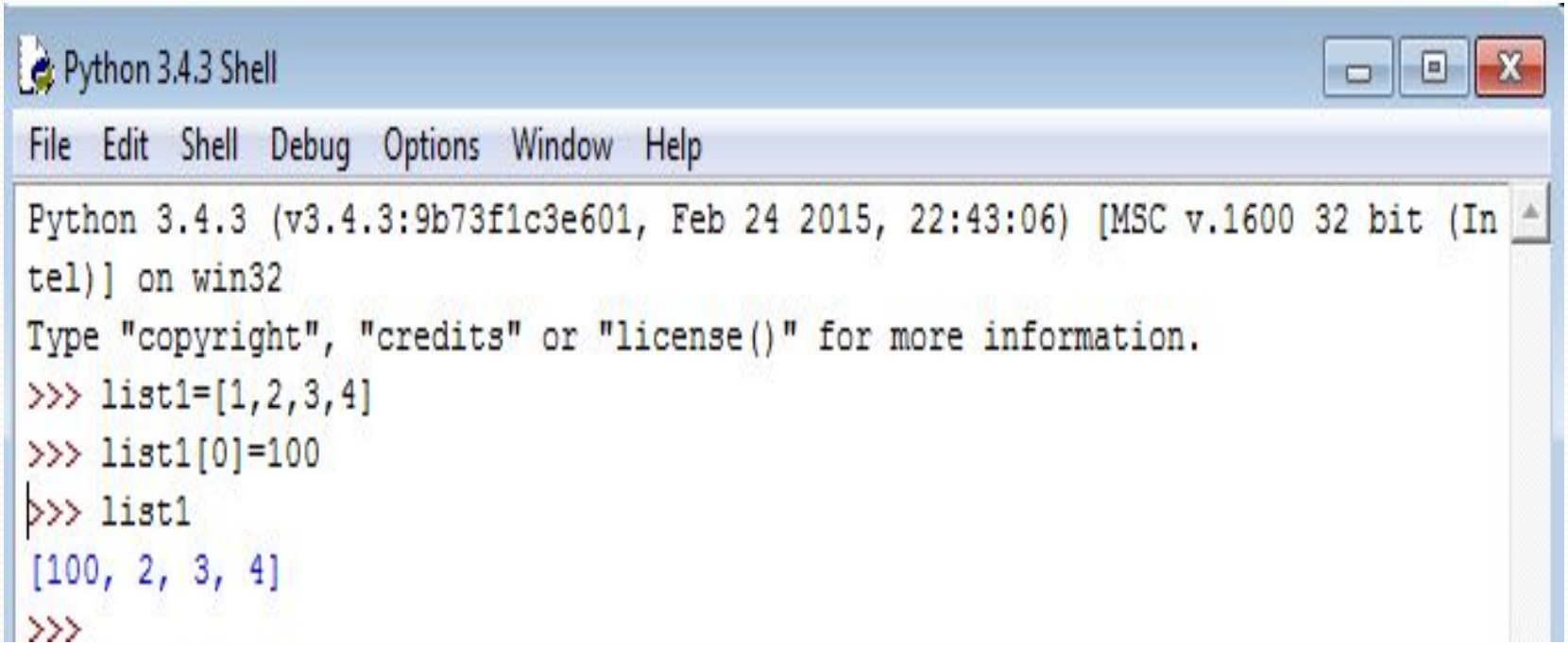
Ln: 10 Col: 4

List are Mutable

- **Types of list:** List is divided into two types
- a. Mutable list
- b. Immutable list
- **Mutable list:** A mutable object can be changed after creating a list. Different languages have different policies on whether string should be mutable. Ruby has mutable.

Mutable list

- When coming to python list are mutable i.e., python will not create a new list but we modify an element in the list.



Immutable list

- **<u>Immutable list</u>:** Instance of this ImmutableList class can be used anywhere a normal python list is expected.

```
>>> a=[1,2,3,4]
>>> a
[1, 2, 3, 4]
>>> b=tuple(a)
>>> b
(1, 2, 3, 4)
>>> a[0]
1
>>> a[0]=2
>>> b
(1, 2, 3, 4)
>>>
```

Traversing a List

- A lot of computations involve processing a sequence one element at a time. The most common pattern is to start at the beginning, select each element in turn, do something to it, and continue until the end. This pattern of processing is called a traversal.

Example

primeexample.py - G:/PythonWorkLog/Practise/primeexample.py (3.6.1)

File Edit Format Run Options Window Help

```
prime_nums = [2, 3, 5, 7, 11]
for num in prime_nums:
    print(num ** 2)
```

Ln: 4 Col: 4

Python 3.6.1 Shell

File Edit Shell Debug Options Window Help

```
Python 3.6.1 (v3.6.1:69c0db5, Mar 21 2017, 17:54:52) [MSC v.1900 32 bit (Intel)]
 on win32
Type "copyright", "credits" or "license()" for more information.
>>>
============= RESTART: G:/PythonWorkLog/Practise/primeexample.py =============
4
9
25
49
121
>>>
```

Ln: 10 Col: 4

List Operations

- **<u>a) Adding Lists</u>:** Lists can be added by using the concatenation operator (+) to join two lists.

Python 3.6.1 Shell

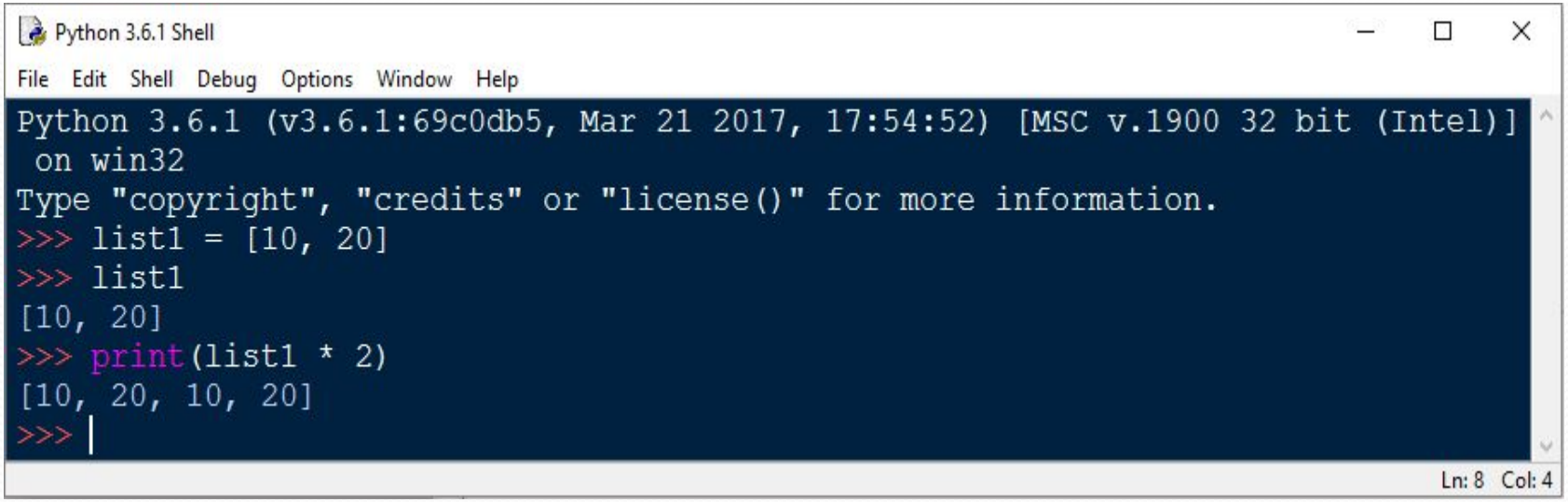
File Edit Shell Debug Options Window Help

```
Python 3.6.1 (v3.6.1:69c0db5, Mar 21 2017, 17:54:52) [MSC v.1900 32 bit (Intel)]
 on win32
Type "copyright", "credits" or "license()" for more information.
>>> list1 = [10, 20]
>>> list1
[10, 20]
>>> list2 = [30, 40]
>>> list2
[30, 40]
>>> list3 = list1 + list2
>>> print(list3)
[10, 20, 30, 40]
>>> |
```

Ln: 12 Col: 4

Replicating Lists

- **<u>b) Replicating Lists:</u>** Replicating means repeating. It can be forming by using (*) operator by specific number of time.



List Slicing

- A subpart of list can be retrieved on the base of index. The subpart is known as list slice.

Python 3.6.1 Shell

File Edit Shell Debug Options Window Help

```
Python 3.6.1 (v3.6.1:69c0db5, Mar 21 2017, 17:54:52) [MSC v.1900 32 bit (Intel)]
 on win32
Type "copyright", "credits" or "license()" for more information.
>>> list1 = [1, 2, 4, 5, 7]
>>> print(list1)
[1, 2, 4, 5, 7]
>>> print(list1[0:2])
[1, 2]
>>> print(list1[4])
7
>>> list1[1] = 9
>>> print(list1)
[1, 9, 4, 5, 7]
>>> |
```

Ln: 13 Col: 4

List Methods

- **Append ():** The append () method adds a single item to the existing list. It doesn't return a new list.
- **<u>Syntax:</u>** list.append (item)

Python 3.4.3 Shell

File Edit Shell Debug Options Window Help

```
Python 3.4.3 (v3.4.3:9b73f1c3e601, Feb 24 2015, 22:43:06) [MSC v.1600 32 bit (In
tel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> a=[1,2,3,4]
>>> a
[1, 2, 3, 4]
>>> a.append(5)
>>> a
[1, 2, 3, 4, 5]
>>> a.append([10,20,30])
>>> a
[1, 2, 3, 4, 5, [10, 20, 30]]
>>>
```

Adding two Lists

- **<u>Adding list to a list:</u>** It is nothing but we have to take two lists i.e., animal and wild_animal. Now some elements are append from one list to another list let us see in below example

```
>>> l1=[10,20,30]
>>> l2=[100,200,300]
>>> l1.append(l2)
>>> l1
[10, 20, 30, [100, 200, 300]]
>>>
```

Extend

- The extend () method takes a single argument (a list) and adds it to the end
- list.extend(list2)
- Here, the elements of list2 are added to the end of list1

```
>>> fruits=["Mango","banana","grapes"]
>>> fruits.extend(["pineapple","orange"])
>>> fruits
['Mango', 'banana', 'grapes', 'pineapple', 'orange']
>>>
```

```
>>> lan1=["english","telugu","tamil"]
>>> lan2=["hindi","french"]
>>> lan1.extend(lan2)
>>> lan1
['english', 'telugu', 'tamil', 'hindi', 'french']
```

INSERT

This method takes two parameters

- Index – position where elements needs to be inserted
- Elements – this is the elements to be inserted in the list

Syntax:

list.insert (index, element)

insert.py - G:/PythonWorkLog/Practise/insert.py (3.6.1)

File Edit Format Run Options Window Help

```
vowel = ['A', 'E', 'I', 'U']
vowel.insert(3, 'O')
print('Updated insert list:', vowel)
```

Ln: 4 Col: 0

```
Updated insert list: ['A', 'E', 'I', 'O', 'U']
```

Remove

- The remove () method takes a single element as an argument and removes it from the list. It does not return any value.
- **Syntax:** list.remove (element)

```
Python 3.6.1 Shell
File  Edit  Shell  Debug  Options  Window  Help
Python 3.6.1 (v3.6.1:69c0db5, Mar 21 2017, 17:54:52) [MSC v.1900 32 bit (Intel)]
 on win32
Type "copyright", "credits" or "license()" for more information.
>>> animal = ["Cat", "Dog", "Rabbit", "Tiger", "Lion", "Dog"]
>>> animal.remove("Dog")
>>> print(animal)
['Cat', 'Rabbit', 'Tiger', 'Lion', 'Dog']
>>> |
Ln: 7  Col: 4
```

Index

- **<u>index () :</u>**method finds the given elements in a list and returns its position. However, if the same element is present more than once, index () method returns its smallest/first position.
- **Syntax:** list.index (element)

index.py - G:/PythonWorkLog/Practise/index.py (3.6.1)

File Edit Format Run Options Window Help

```
vowel = ['A', 'E', 'I', 'O', 'I' ,'U']

index = vowel.index('E')
print("The index of E:", index)

index = vowel.index('I')
print("The index of I:", index)

index = vowel.index('B')
print("The index of B:", index)
```

Ln: 7 Col: 31

Continue….

The index () method returns the index of the element in the list. If not found, it raises a ValueError exception indicating the element is not in the list.

```
Python 3.6.1 Shell
File  Edit  Shell  Debug  Options  Window  Help
Python 3.6.1 (v3.6.1:69c0db5, Mar 21 2017, 17:54:52) [MSC v.1900 32 bit (Intel)]
 on win32
Type "copyright", "credits" or "license()" for more information.
>>> vowel = ['A', 'E', 'I', 'O', 'I', 'U']
>>>
================ RESTART: G:/PythonWorkLog/Practise/index.py ================
The index of E: 1
The index of I: 2
Traceback (most recent call last):
  File "G:/PythonWorkLog/Practise/index.py", line 9, in <module>
    index = vowel.index('B')
ValueError: 'B' is not in list
>>> |
                                                                  Ln: 12  Col: 4
```

Count

- **Count ()** method count how many times an element has occurred in a list and returns it count.
- **Syntax:** list.count(element)

count.py - G:/PythonWorkLog/Practise/count.py (3.6.1)

File Edit Format Run Options Window Help

```
vowels = ['A', 'E', 'I', 'O', 'I', 'U']

count = vowels.count('I')
print('Count Of I is:',count)

count = vowels.count('P')
print('Count of P is:',count)
```

Ln: 6 Col: 23

```
Count Of I is: 2
Count of P is: 0
```

Pop

- The pop () method takes a single argument (index) and removes the element present at that index from the list
- If no parameter is passed, the <u>default index -1 is passed as an argument which returns the last element.</u>

```
>>> list1=[11,12,13,14,15]
>>> list1.pop()
15
>>> list1
[11, 12, 13, 14]
>>> list1.pop(2)
13
>>> list1
[11, 12, 14]
>>>
```

Reverse

- The reverse () function doesn't return any value. It only reverses the elements and updates the list
- **<u>Syntax</u>**: list.reverse ()

reverse.py - G:/PythonWorkLog/Practise/reverse.py (3.6.1)

File Edit Format Run Options Window Help

```
os = ['Windows', 'MacOS', 'Linux']

print("Original list is:", os)

os.reverse()
print("Updated list is:", os)
```

Ln: 6 Col: 29

Python 3.6.1 Shell

File Edit Shell Debug Options Window Help

```
Python 3.6.1 (v3.6.1:69c0db5, Mar 21 2017, 17:54:52) [MSC v.1900 32 bit (Intel)]
 on win32
Type "copyright", "credits" or "license()" for more information.
>>>
=============== RESTART: G:/PythonWorkLog/Practise/reverse.py ===============
Original list is: ['Windows', 'MacOS', 'Linux']
Updated list is: ['Linux', 'MacOS', 'Windows']
>>>
```

Ln: 7 Col: 4

Sort

- **<u>The sort ()</u>** method sorts the elements of a given list in a specific order Ascending or Descending.
- **Syntax:** list.sort (key= ..., reverse= ....)

sort.py - G:/PythonWorkLog/Practise/sort.py (3.6.1)

File Edit Format Run Options Window Help

```
vowels = ['E', 'A', 'U', 'a', 'O', 'I', 'o']

vowels.sort()
print("Sorted list:", vowels)
```

Ln: 4 Col: 29

```
Sorted list: ['A', 'E', 'I', 'O', 'U', 'a', 'o']
```

How to sort in descending order?

- Sort () method accepts a reverse parameter as an optional argument
- Syntax:list.sort (reverse=True)

sort1.py - G:/PythonWorkLog/Practise/sort1.py (3.6.1)

File Edit Format Run Options Window Help

```
vowels = ['e', 'a', 'u', 'o', 'i']

vowels.sort(reverse=True)
print("Sorted list in Descending:", vowels)

vowels.sort(reverse=False)
print("Sorted list in Ascending:", vowels)
```

Ln: 8 Col: 0

```
Sorted list in Descending: ['u', 'o', 'i', 'e', 'a']
Sorted list in Ascending: ['a', 'e', 'i', 'o', 'u']
```

Copy

- The problem with copying the list in this way is that if modify the new list, the old list is also modified
- Syntax: new_list = old_list

copy.py - G:/PythonWorkLog/Practise/copy.py (3.6.1)

File Edit Format Run Options Window Help

```
old_list = [1, 2, 3]
new_list = old_list
new_list.append('a')

print("New list:", new_list)
print("Old list:", old_list)
```

Ln: 6 Col: 28

```
New list: [1, 2, 3, 'a']
Old list: [1, 2, 3, 'a']
```

Clear

- The clear () method only empties the given list doesn't take any parameters
- <u>Syntax:</u> list.clear ()

Python 3.6.1 Shell

File Edit Shell Debug Options Window Help

```
Python 3.6.1 (v3.6.1:69c0db5, Mar 21 2017, 17:54:52) [MSC v.1900 32 bit (Intel)]
 on win32
Type "copyright", "credits" or "license()" for more information.
>>> list1 = ["A", "B", "C", "D", "E"]
>>> list1
['A', 'B', 'C', 'D', 'E']
>>> list1.clear()
>>> list1
[]
>>>
```

Ln: 4 Col: 9

Deleting Elements

- Deleting multiple items from a list is not directly possible in one command in python.

```
>>> li=[1,6,2,73]
>>> del(li[2])
>>> li
[1, 6, 73]
```

Lambda function

- **Anonymous function** is a function that is defined without a name.
- **Normal functions** are defined using the **def** keyword and **Python anonymous functions** are defined using the **lambda** keyword.
- **Lambda Syntax**: lambda arguments: expression
- Lambda functions can have any number of arguments but only one expression. The expression is evaluated and returned. Lambda functions can be used wherever function objects are required.

Example

```
Python 3.6.3 Shell
File  Edit  Shell  Debug  Options  Window  Help
Python 3.6.3 (v3.6.3:2c5fed8, Oct  3 2017, 17:26:49) [MSC v.1900 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> double=lambda x:x*2
>>> print(double(5))
10
>>>
```

In the above program, lambda x: x * 2 is the lambda function. Here x is the argument and x * 2 is the expression that gets evaluated and returned

Map

- The map () function applies a given to function to each item of an iterable and returns a list of the results.

```
>>> y=lambda x:x*2
>>> print(list(map(y,list(range(1,4)))))
[2, 4, 6]
```

Example 2

```
>>> print(list(map(lambda x:x+1,[12,13,14,15])))
[13, 14, 15, 16]
```

Reduce

- Reduce is a really useful function for performing some computation on a list and returning the result. It applies a rolling computation to sequential pairs of values in a list.

```
Python 3.6.3 Shell
File  Edit  Shell  Debug  Options  Window  Help
Python 3.6.3 (v3.6.3:2c5fed8, Oct  3 2017, 17:26:49) [MSC v.1900 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> from functools import reduce
>>> p=reduce((lambda a,b:a+b),[1,2,3,4,5])
>>> p
15
>>>
```

Filter

- The filter() method constructs an iterator from elements of an iterable for which a function returns true
- the filter() method filters the given iterable with the help of a function that tests each element in the iterable to be true or not
- <u>The syntax of filter() method is</u>:
- filter(function, iterable)

Example

filter (1).py - C:\Users\student\Desktop\filter (1).py (3.4.3)

File Edit Format Run Options Window Help

```
a=['a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s','t','u','v','w','x','y','z']

def fil(a):
    v=['a','e','i','o','u']
    if (a in v):
        return True
    else:
        return False
ff=filter(fil,a)
print("The filtered vowels are:")
print(list(ff))
```

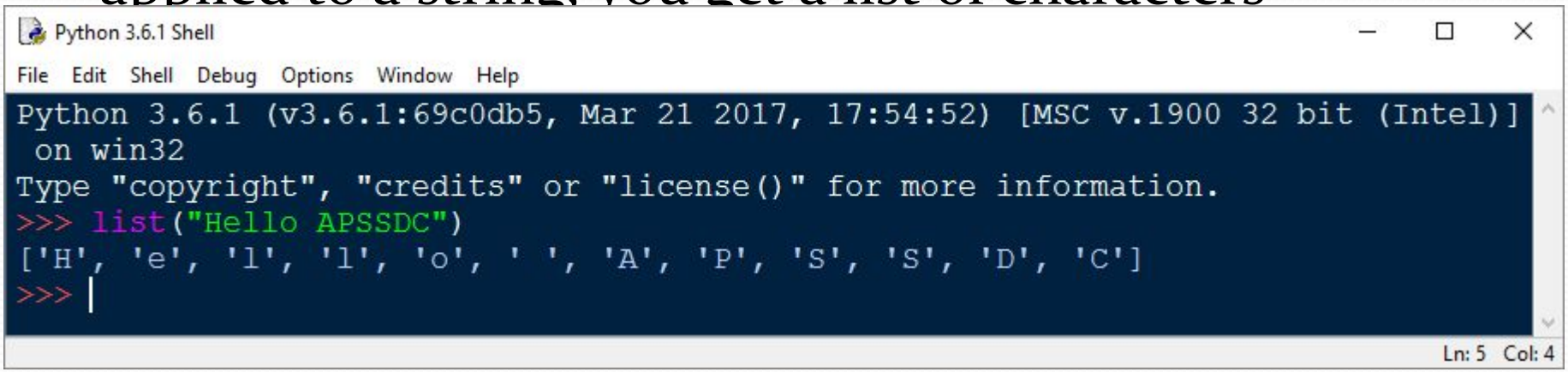
Python 3.4.3 Shell

File Edit Shell Debug Options Window Help

```
Python 3.4.3 (v3.4.3:9b73f1c3e601, Feb 24 2015, 22:43:06) [MSC v.1600 32 bit (In
tel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> ================================ RESTART ================================
>>>
The filtered vowels are:
['a', 'e', 'i', 'o', 'u']
```

Lists and Strings

- Python has several tools which combine lists of strings into strings and separate strings into lists of strings.
- The list command takes a sequence type as an argument and creates a list out of its elements. When applied to a string, you get a list of characters



Split

- The split method invoked on a string and separates the string into a list of strings, breaking it apart whenever a substring called the delimiter occurs. The default delimiter is whitespace, which includes spaces, tabs, and newlines

Python 3.6.1 Shell

File Edit Shell Debug Options Window Help

```
Python 3.6.1 (v3.6.1:69c0db5, Mar 21 2017, 17:54:52) [MSC v.1900 32 bit (Intel)]
 on win32
Type "copyright", "credits" or "license()" for more information.
>>> str1 = "Hai everyone Welcome to APSSDC"
>>> str1.split()
['Hai', 'everyone', 'Welcome', 'to', 'APSSDC']
>>> |
```

Ln: 6 Col: 4

Continue…

- Here we have 'h' as the delimiter

```
Python 3.6.1 Shell
File  Edit  Shell  Debug  Options  Window  Help
Python 3.6.1 (v3.6.1:69c0db5, Mar 21 2017, 17:54:52) [MSC v.1900 32 bit (Intel)]
 on win32
Type "copyright", "credits" or "license()" for more information.
>>> str1 = "hai hello how are you"
>>> str1.split('h')
['', 'ai ', 'ello ', 'ow are you']
>>> |
Ln: 6  Col: 4
```

Notice that the delimiter doesn't appear in the list. The join method does approximately the opposite of the split method

Join

It takes a list of strings as an argument and returns a string of all the list elements joined together

Python 3.6.1 Shell

File Edit Shell Debug Options Window Help

```
Python 3.6.1 (v3.6.1:69c0db5, Mar 21 2017, 17:54:52) [MSC v.1900 32 bit (Intel)]
 on win32
Type "copyright", "credits" or "license()" for more information.
>>> str1 = ' '
>>> ' '.join(['hai','hello','welcome','to','APSSDC'])
'hai hello welcome to APSSDC'
>>> |
```

Ln: 6 Col: 4

```
>>> '************'.join(['hello','everyone','are','you','fine'])
'hello************everyone************are************you************fine'
```

Continue…..

- The string value on which the join method is invoked acts as a separator that gets placed between each element in the list in the returned string.

```
Python 3.6.1 Shell
File  Edit  Shell  Debug  Options  Window  Help
Python 3.6.1 (v3.6.1:69c0db5, Mar 21 2017, 17:54:52) [MSC v.1900 32 bit (Intel)]
 on win32
Type "copyright", "credits" or "license()" for more information.
>>> str1 = ' '
>>> '**'.join(['hai','hello','welcome','to','APSSDC'])
'hai**hello**welcome**to**APSSDC'
>>> |
Ln: 6  Col: 4
```

The separator can also be the empty string

```
Python 3.6.1 Shell
File  Edit  Shell  Debug  Options  Window  Help
Python 3.6.1 (v3.6.1:69c0db5, Mar 21 2017, 17:54:52) [MSC v.1900 32 bit (Intel)]
 on win32
Type "copyright", "credits" or "license()" for more information.
>>> ''.join(['hai','hello','welcome','to','APSSDC'])
'haihellowelcometoAPSSDC'
>>> |
Ln: 5  Col: 4
```