

STRING

Pratik Joshi



String in Python

- String is sequence of character.
- Python doesn't have character data type a single character is a string with a length of 1.
- String can be created using single quote or double quote.
- Strings are Array of characters and we can loop in a string.
- Escape character:
 - `\n` : New line
 - `\t` : tab
 - `\\` : backslash
 - `\r` : carriage return (Move the cursor to start of line)
 - `\"` : escape double quote
- We can access individual characters using indexing and a range of characters using **slicing**.

Slicing

Slicing is used to return a range of characters by using slicing operator (:)

B	H	U	B	A	N	E	S	W	A	R
0	1	2	3	4	5	6	7	8	9	10
-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

Syntax: string[start:end:step]

- From start to end-1
- Python allows negative indexing, index of -1 represent last character.

Slicing Examples

```
city='BHUBANESWAR'
```

```
print(city[0])
```

```
print(city[5])
```

```
print(city[1:5])
```

```
print(city[3:])
```

```
print(city[:5])
```

```
print(city[:])
```

```
print(city[5:2])
```

```
print(city[0:50])
```

```
print(city[-1])
```

```
print(city[-2])
```

```
print(city[5:-2])
```

```
print(city[-5:-2])
```

```
print(city[-5:10])
```

```
print(city[0:9:2])
```

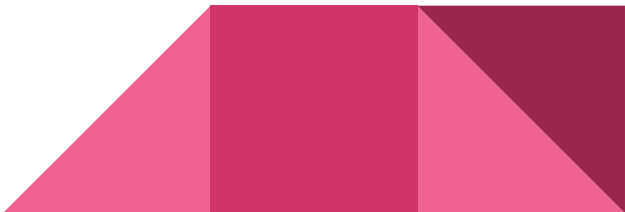
```
print(city[9:2:-1])
```

```
print(city[2:-2:2])
```

```
print(city[-2:2:-2])
```

```
print(city[-1::-1])
```

Pratik Joshi



String Modification

- String is immutable so elements of string cannot be changed.
- We can delete the entire string by del string_name.

String Operation:

- Concatenation using + and *
 - Str1+str2, str*3
- String Iteration using loop

```
for i in str:  
    print(i)
```

Pratik Joshi



String Methods

- `len(city)`
- `'India'.upper()`
- `'CAptial'.lower()`
- `strip()` : remove white space from beginning or the end

- `split()`

E.g

```
s1="Hey There"
```

```
s1.split()
```

```
['Hey', 'There']
```

```
s2="apple,banana,pineapple"
```

```
s2.split(',')
```

```
['apple', 'banana', 'pineapple']
```

- `find()`

```
'jungle'.find('ng')
```

```
2
```

- `replace()`

```
'good morning'.replace('morning','afternoon')
```

```
'good afternoon'
```

LIST

Pratik Joshi



List in Python

- Built-in datatype used to store collections of data
- Mutable
- Ordered
- Allows Duplicates
- List-items can be of any data types

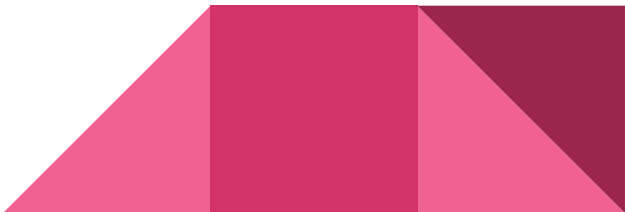
List creation: by placing all the items inside square brackets `[]`, separated by ,

```
list1= [1, "Hello", 3.4]
```

Accessing List items:

- List Index
- Negative Indexing
- Slicing

Pratik Joshi



Modify List Items

Change Single Item:

```
fruits = ["apple", "banana", "cherry"]  
fruits[2] = 'mango'
```

Change Range of items:

```
fruits = ["apple", "banana", "mango"]  
fruits[1:3] = ["watermelon", "papaya"]
```

Insert and append:

```
fruits = ["apple", "banana", "papaya"]  
fruits.insert(2, "pineapple")  
fruits.append("orange")
```

```
state = ['o', 'd', 'i', 's', 'h', 'a']
```

```
# delete one item
```

```
del state[2]
```

```
# delete multiple items
```

```
del state[1:5]
```

```
# delete entire list
```

```
del state
```



Modify List Items (Cont...)

```
fruits = ["apple", "banana", "papaya"]
```

```
#removing index
```

```
fruits.remove("papaya")
```

```
#removing index
```

```
fruits.pop(1)
```

```
fruits.pop(1)
```

```
#Clearing list
```

```
fruits.clear()
```

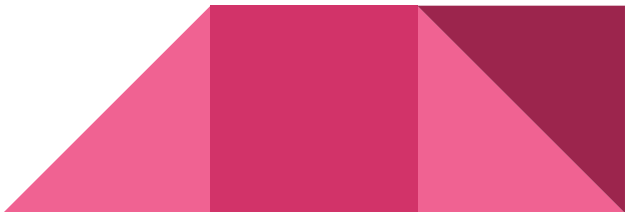
```
# Concatenating and Repeating
```

```
odd = [1, 3, 5]
```

```
print(odd + [2, 4, 6])
```

```
print(odd * 3)
```

Pratik Joshi



List Comprehension

Used to shorter the syntax when a new list is to be created based on the values of an existing list.

Syntax:

```
newlist = [expression for item in list if condition == True]
```

E.g:

```
centry=['i','n','d','i','a']
```

```
centry_new=[i.upper() for i in centry ]
```

```
print(centry_new)
```

```
['I', 'N', 'D', 'I', 'A']
```

```
lst=[1,2,3,4,5,6,7,8,9,10]
```

```
even=[i for i in lst if i%2==0]
```

```
print(even)
```

```
[2, 4, 6, 8, 10]
```



List Comprehension VS Loop

```
import time
```

```
n=10**6
```

```
begin = time.time()
```

```
# in loop
```

```
result = []
```

```
for i in range(n):
```

```
    result.append(i ** 2)
```

```
end = time.time()
```

```
print('Time taken for loop:', round(end - begin, 2))
```

```
n=10**6
```

```
begin_lc = time.time()
```

```
# in list comprehension
```

```
res=[i ** 2 for i in range(n)]
```

```
end_lc = time.time()
```

```
print('Time taken for list_comprehension:', round(end_lc - begin_lc, 2))
```

Time taken for loop: 0.34

Time taken for list_comprehension: 0.28 #Faster than Loop

List Functions

```
sort():
```

```
str_list = ["orange", "mango", "kiwi", "pineapple", "banana"]
```

```
str_list.sort()
```

```
num_list = [101, 49, 6, 99, 25]
```

```
num_list.sort()
```

```
num_list.sort(reverse=True) #Sort in descending order
```

```
copy()    : new_list=num_list.copy()    #copy the entire list
```

```
len()     : len(new_list)               #return length of list
```

```
count()   : new_list.count(99)          #return count of passed argument
```

```
reverse() : new_list.reverse()          #Reverse the list
```

```
index()   : new_list.index(25)          #return index of first matched
```

Hands-On 2

1. WAP to find sum of all items in list.
2. WAP to find largest number in a numeric list.
3. WAP to find average value of a list.
4. WAP to calculate total number of integer in a list.
5. Create a list with 1-100 and create two separate list with Odd and Even numbers using List Comprehension.
6. Reverse a List using slicing technique.
7. WAP to remove duplicates in a given list. [10,10,10,20,20,30,40,40,50,70,90]
output: [10,20,30,40,50,70,90]
8. WAP to accept user's sentence and print the middle word if length is odd and print middle two if length of sentence is even.
User Input: "I love my India"
Expected Op: love, my
User Input: "India is largest democracy in the world"
Expected Op: democracy