String Indexing

In [1]:
```
a = 'Samosa Pakorra'
print(a)
```
```
Samosa Pakorra
```

In [2]:
```
a[12]
```
Out[2]:
```
'r'
```

In [3]:
```
a[13]
```
Out[3]:
```
'a'
```

In [4]:
```
# -1 start printing from last
print(a[-1])
print(a[-2])
```
```
a
r
```

In [5]:
```
# length of string
len(a)
```
Out[5]:
```
14
```

In [6]:
```
# start from zero upto 5, But 5 is exclusive mean 5 no will not count
print(a[0:5]) #Samos
print(a[:3])   # Last digit is not included
```
```
Samos
Sam
```

In [7]:
```
print(a[ : -10])
print(a[ -6 : -10])
```
```
Samo
```

String Methods

In [8]:
```
food = 'baryani'
food
```
Out[8]:
```
'baryani'
```

In [9]:
```
# find length of baryani
len(food)
```

```
Out[9]: 7
```

```
In [10]:
# it will capitalize the words first letter
food.capitalize()
```

```
Out[10]: 'Baryani'
```

```
In [11]:
sentence = 'we are Learning Python with Baba Aammar'
# capitalize first letter and if already then it become small
sentence.capitalize()
```

```
Out[11]: 'We are learning python with baba aammar'
```

```
In [12]:
# uppercase letter
food.upper()
```

```
Out[12]: 'BARYANI'
```

```
In [13]:
#lower case letter
food.lower()
```

```
Out[13]: 'baryani'
```

```
In [14]:
# we can replace letter inside string
food.replace('ba', 'sha')
```

```
Out[14]: 'sharyani'
```

```
In [15]:
# counting a specific alphabet in a string
course = 'Learning python with baba Aammar'
course.count('n')
```

```
Out[15]: 3
```

- **Find index in a string**

```
In [16]:
#find index of a letter in a string
course.find('y')
```

```
Out[16]: 10
```

Split Strings

```
In [17]:
menu_list = 'I love chai, Karrahi, baryani, Rayata'
menu_list.split(' ') # split if there is space
```

```
Out[17]: ['I', 'love', 'chai,', 'Karrahi,', 'baryani,', 'Rayata']
```

```
In [18]:
# split(): split the string and convert it into List
a = menu_list.split(',')
print(type(a))
```

```
<class 'list'>
```

Tuples

- Order collections of elements
- enclosed in () round braces / parenthesis
- Different kind of elements can be stored
- Once value is stored in Tuple it can't be change [immurtable]

In [19]:
```
tup1 = (3, 'python', 3.14, False)
tup1
```
Out[19]:
```
(3, 'python', 3.14, False)
```

In [20]:
```
tup1.count('o')
```
Out[20]:
```
0
```

In [21]:
```
# Type of Tuple
type(tup1)
```
Out[21]:
```
tuple
```

Indexing in Tuple

In [22]:
```
tup1[1]
```
Out[22]:
```
'python'
```

In [23]:
```
tup1[-1]
```
Out[23]:
```
False
```

In [24]:
```
tup1[-2]
```
Out[24]:
```
3.14
```

In [25]:
```
tup1[0:2]
```
Out[25]:
```
(3, 'python')
```

In [26]:
```
# print 3 time the tup1
tup1 * 3
```
Out[26]:
```
(3, 'python', 3.14, False, 3, 'python', 3.14, False, 3, 'python', 3.14, False)
```

In [27]:
```
# [03:14:33]  why tup1 + 2 doesn't work,
#  because it can only concatenate tuple to tuple
# tup1 + 2
```

In [28]:
```
# Concatinating tuple (to add two or more tupe)
tup2 = (34, True, 2.2, 'DL')
tup2
tup1 + tup2
```

Out[28]:
```
(3, 'python', 3.14, False, 34, True, 2.2, 'DL')
```

In [29]:
```
tup3 = (2, 5, 7, 7, 90, 12)
min(tup3)
```

Out[29]:
```
2
```

In [30]:
```
max(tup3)
```

Out[30]:
```
90
```

In [31]:
```
tup3 *3
```

Out[31]:
```
(2, 5, 7, 7, 90, 12, 2, 5, 7, 7, 90, 12, 2, 5, 7, 7, 90, 12)
```

* three deshes used for line

List

- Order collections of elements
- Enclosed in [] square brackets
- Mutatable, mean we can change elements of lists

In [32]:
```
list1 = [23, 'Codanics', 0.2, False]
list1
```

Out[32]:
```
[23, 'Codanics', 0.2, False]
```

In [33]:
```
# find type
type(list1)
```

Out[33]:
```
list
```

In [34]:
```
#find the length of string
len(list1)
```

Out[34]:
```
4
```

In [35]:
```
list2 = [1,22,67, 90, 'We are learning python', {1: 'Codanics'}]
list2
```

```
Out[35]: [1, 22, 67, 90, 'We are learning python', {1: 'Codanics'}]
```

```
In [36]: # list concatenation
         list1 + list2
```

```
Out[36]: [23,
          'Codanics',
          0.2,
          False,
          1,
          22,
          67,
          90,
          'We are learning python',
          {1: 'Codanics'}]
```

```
In [37]: # multiply list
         list1 * 2
```

```
Out[37]: [23, 'Codanics', 0.2, False, 23, 'Codanics', 0.2, False]
```

List built_in functions

```
In [44]: list3 = [45, 3, 67, 200, 34, 2, 6, 88]
         list3
```

```
Out[44]: [45, 3, 67, 200, 34, 2, 6, 88]
```

```
In [45]: # append() add element at the end
         list3.append(70)
         list3
```

```
Out[45]: [45, 3, 67, 200, 34, 2, 6, 88, 70]
```

```
In [47]: list3.copy()  # make copy of the list
         list3
```

```
Out[47]: [45, 3, 67, 200, 34, 2, 6, 88, 70]
```

```
In [50]: # extend the current list with other one
         list3.extend(list2)
         list3
```

```
Out[50]: [45,
          3,
          67,
          200,
          34,
          2,
          6,
          88,
          70,
          1,
          22,
          67,
          90,
          'We are learning python',
          {1: 'Codanics'},
          1,
          22,
          67,
          90,
          'We are learning python',
          {1: 'Codanics'},
          1,
          22,
          67,
          90,
          'We are learning python',
          {1: 'Codanics'}]
```

In [52]:
```
list3.index(22) #index of a specific value in list
```

Out[52]:
```
10
```

In [54]:
```
# insert(): insert value at a specific position
list3.insert(0, 88)
list3
```

Out[54]:
```
[88,
 45,
 3,
 67,
 200,
 34,
 2,
 6,
 88,
 70,
 1,
 22,
 67,
 90,
 'We are learning python',
 {1: 'Codanics'},
 1,
 22,
 67,
 90,
 'We are learning python',
 {1: 'Codanics'},
 1,
 22,
 67,
 90,
 'We are learning python',
 {1: 'Codanics'}]
```

In [56]:
```
list3.remove(88)
list3
```

Out[56]:
```
[45,
 3,
 67,
 200,
 34,
 2,
 6,
 88,
 70,
 1,
 22,
 67,
 90,
 'We are learning python',
 {1: 'Codanics'},
 1,
 22,
 67,
 90,
 'We are learning python',
 {1: 'Codanics'},
 1,
 22,
 67,
 90,
 'We are learning python',
 {1: 'Codanics'}]
```

List.count()

- Count() method return of many times an object is occure in a list.

- e.g: list1.count(object)

here object is the thing whose count is to be return.

In [62]:
```
list1 = [3, 4, 3, 5, 6, 4, 4, 8, 6, 6, 5]
list1.count(5) # it count how many time 4 occurs in the list
```

Out[62]:
```
2
```

In [63]:
```
list_name = ['a', 'r', 'a', 'z', 'r', 'a', 'z', 'z']
list_name.count('z') # it count how many time 'z' occurs in the list
```

Out[63]:
```
3
```

- ***Expection***
- If more than one argument is passed into count() it return TypeError()

In [65]:
```
list_name.count('a', 'z')
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
~\AppData\Local\Temp/ipykernel_4376/2490571529.py in <module>
----> 1 list_name.count('a', 'z')

TypeError: list.count() takes exactly one argument (2 given)
```

Dictionaries

- An unordered collection of elements
- enclosed in {} curly brackets
- key and value pair

- Mutatable, we can change elements of dict

In [68]:
```
food1 = {'Samosa': 10, 'Salad' : 30, 'Raita': 10}
food1
```
Out[68]:
```
{'Samosa': 10, 'Salad': 30, 'Raita': 10}
```

In [71]:
```
# print keys of the dictionaries
keys = food1.keys()
keys
```
Out[71]:
```
dict_keys(['Samosa', 'Salad', 'Raita'])
```

In [74]:
```
# print keys of the dictionaries
values = food1.values()
values
```
Out[74]:
```
dict_values([10, 30, 10])
```

In [76]:
```
# Update value
food1['Raita'] = 35
food1
```
Out[76]:
```
{'Samosa': 10, 'Salad': 30, 'Raita': 35}
```

In [86]:
```
food1.update({'Brayani': 100} )
food1
```
Out[86]:
```
{'Samosa': 10, 'Salad': 30, 'Raita': 35, 'Brayani': 100}
```

In [90]:
```
food1.update({'Raita': 15} )
food1
```
Out[90]:
```
{'Samosa': 10, 'Salad': 30, 'Raita': 15, 'Brayani': 100}
```

In [92]:
```
food2 = {1 : 'Baba Ammar', 2 : 'Rehman', 3 : 'Machine Learning ka Chilla'}
food2
```
Out[92]:
```
{1: 'Baba Ammar', 2: 'Rehman', 3: 'Machine Learning ka Chilla'}
```

In [96]:
```
# Concatenation of two dictt
food1.update(food2)
food1
```
Out[96]:
```
{'Samosa': 10,
 'Salad': 30,
 'Raita': 15,
 'Brayani': 100,
 1: 'Baba Ammar',
 2: 'Rehman',
 3: 'Machine Learning ka Chilla'}
```

In [100…

```
# make copy
food1.copy()
food1
```

Out[100…

```
{'Samosa': 10,
 'Salad': 30,
 'Raita': 15,
 'Brayani': 100,
 1: 'Baba Ammar',
 2: 'Rehman',
 3: 'Machine Learning ka Chilla'}
```

In [109…

```
a =food1.fromkeys('Salad')
a
```

Out[109…

```
{'S': None, 'a': None, 'l': None, 'd': None}
```

In [114…

```
# get key and return its value
g = food1.get(1)
g
```

Out[114…

```
'Baba Ammar'
```

In [129…

```
# it returns list of Tuple of key value pair
i = food1.items()
i
```

Out[129…

```
dict_items([('Samosa', 10), ('Salad', 30), ('Raita', 15), ('Brayani', 100), (1, 'Bab
a Ammar'), (2, 'Rehman')])
```

In [156…

```
# pop(): pop the value of the key that we give
print(food1)
x = food1.pop(1)
print(x)
```

```
{'Salad': 30, 'Raita': 15, 'Brayani': 100, 1: 'Baba Ammar'}
Baba Ammar
```

In [ ]:

```
food1.clear() # clear the dictionaries
```

Set

- Unordered and Unindexed
- enclosed in {} curely braces
- No duplicate value is allowed

In [123…

```
set1 = {2, 5, 'Toyota', 'Machine Learning', True}
set1
```

Out[123…

```
{2, 5, 'Machine Learning', 'Toyota', True}
```

In [125…

```
set1.add('Deep Learning')
set1
```

Out[125…

```
{2, 5, 'Deep Learning', 'Machine Learning', 'Toyota', True}
```

In [127…

```
set1.add(2) # 2 is already there, so it will not update the set
set1
```

Out[127…

```
{2, 5, 'Deep Learning', 'Machine Learning', 'Toyota', True}
```

In [130…

```
# Make copy of the set
set1.copy()
set1
```

Out[130…

```
{2, 5, 'Deep Learning', 'Machine Learning', 'Toyota', True}
```

In [141…

```
# discard()
set1.discard(5)
set1
```

Out[141…

```
{'Deep Learning', 'Toyota', True}
```

In [145…

```
set3 = {4, 7, 2, 5, 2, 8}
set4 = {6, 10, 2, 8}
```

In [147…

```
# find the difference b/w two sets
set3.difference(set4)
```

Out[147…

```
{4, 5, 7}
```

In [157…

```
# Intersection()
set3.intersection(set4)
```

Out[157…

```
{2, 8}
```

In [158…

```
# isdisjoint()
set3.isdisjoint(set4)
```

Out[158…

```
False
```

In [160…

```
set4.issubset(set3)
```

Out[160…

```
False
```

In [161…

```
set3
```

Out[161…

```
{2, 4, 5, 7, 8}
```

In [164…

```
# pop out the first element from the set
set3.pop()
set3
```

Out[164…

```
{5, 7, 8}
```

In [166…

```
# Union() of two sets
set3.union(set4)
```

Out[166…

```
{2, 5, 6, 7, 8, 10}
```

In [175…

```
# clear the set
set4.clear()
set4
```

Out[175…

```
set()
```

In [ ]: