Cheatography

Python List, dictionaries, tuple, string methods Cheat Sheet

by srinivas.ram via cheatography.com/183208/cs/38138/

Python Lists

List

To Create a List:

```
thislist = ["apple", "banana", "cherry"]
print(thislist)
```

output::

```
['apple', 'banana', 'cherry']
```

A list can contain different data types::

Example::

```
list1 = ["abc", 34, True, 40, "male"]
```

What is the data type of a list?

```
mylist = ["apple", "banana", "cherry"]
print(type(mylist))
```

output:: **<class 'list'>**

List items can be accessed by referring to the index number:

```
thislist = ["apple", "banana", "cherry"]
print(thislist[1])
```

output:: **banana**

To change the value of a specific item, refer to the index number:

```
thislist = ["apple", "banana", "cherry"]
thislist[1] = "blackcurrant"
print(thislist)
```

output:: **['apple', 'blackcurrant', 'cherry']**

To add an item to the end of the list, use the append() method:

```
thislist = ["apple", "banana", "cherry"]
thislist.append("orange")
print(thislist)
```

output:: **['apple', 'banana', 'cherry', 'orange']**

Python Lists (cont)

To append elements from another list to the current list, use the **extend()** method.

```
list_A = ["apple", "banana", "cherry"]
list_B = ["mango", "pineapple", "papaya"]
list_A.extend(list_B)
print(list_A)
```

output::

```
['apple', 'banana', 'cherry', 'mango', 'pinea-
pple', 'papaya']
```

The **remove()** method removes the specified item.

```
list_A = ["apple", "banana", "cherry"]
list_A.remove("banana")
print(list_A)
```

output:: **['apple', 'cherry']**

The **pop()** method removes the specified index
with out idex it will remove last item

```
list_A = ["apple", "banana", "cherry"]
list_A.pop(1)
print(list_A)
```

output:: **['apple', 'cherry']**

The **del keyword removes the specified index
Also delete the list completely

```
thislist = ["apple", "banana", "cherry"]
del thislist[0]
print(thislist)
```

output::**['banana', 'cherry']**

The **clear()** method empties the list.

```
thislist = ["apple", "banana", "cherry"]
thislist.clear()
print(thislist)
```

output::**[]**

loop through the list items by using a for loop:

Python Lists (cont)

```
thislist = ["apple", "banana", "cherry"]
for x in thislist:
print(x)
```

output::

```
apple
banana
cherry
```

loop through the list items by referring to their index number.
Use the **range()** and **len()** functions to create a suitable iterable.

```
thislist = ["apple", "banana", "cherry"]
for i in range(len(thislist)):
print(thislist[i])
```

output::

```
apple
banana
cherry
```

loop through the list items by using a while loop.

```
thislist = ["apple", "banana", "cherry"]
i = 0
while i < len(thislist):
print(thislist[i]) i = i + 1
```

output ::

```
apple
banana
cherry
```

List Comprehension offers the shortest syntax for looping through lists:

```
thislist = ["apple", "banana", "cherry"]
[print(x) for x in thislist]
```

output::

```
apple
banana
cherry
```

List comprehension offers a shorter syntax when you want to create a new list based on the values of an existing list.

Python Lists (cont)

Without list comprehension

fruits = ["apple", "banana", "cherry", "kiwi", "-mango"]
newlist = []
for x in fruits:
if "a" in x:
newlist.append(x)
print(newlist)

With list comprehension you can do all that with only one line of code

fruits = ["apple", "banana", "cherry", "kiwi", "-mango"]
newlist = [x for x in fruits if "a" in x]
print(newlist)

sort() method that will sort the list alphanumerically, ascending, by default:

```
thislist = ["orange", "mango", "kiwi", "pineapple", "banana"]
thislist.sort()
print(thislist)
```

output::
['banana', 'kiwi', 'mango', 'orange', 'pineapple']

To make a copy, one way is to use the built-in List method **copy()**.

```
thislist = ["apple", "banana", "cherry"]
mylist = thislist.copy()
print(mylist)
```

output::**['apple', 'banana', 'cherry']**

List count() Method

Return the number of times the value "cherry"
appears in the fruits list:

Python Lists (cont)

```
fruits = ['apple', 'banana', 'cherry']
x = fruits.count("cherry")
```

output::
1

There are some list methods that will change the order, but in general: the order of the items will not change.

Dictionaries in Python

Dictionaries are used to store data values in **key:value pairs.**
It is a collection which is ordered*, changeable and do not allow duplicates.

Create and print a dictionary:

```
thisdict = { "brand": "Ford", "model": "Mustang", "year": 1964 }
print(thisdict)
```

output
{'brand': 'Ford', 'model': 'Mustang', 'year': 1964}

You can access the items of a dictionary by referring to
its **key name**, inside square brackets:

```
thisdict = { "brand": "Ford", "model": "Mustang", "year": 1964 }
x = thisdict["model"]
print(x)
```

output::**Mustang**

The **keys() method** will return a list of all the keys in the dictionary.

```
thisdict = { "brand": "Ford", "model": "Mustang", "year": 1964 }
x = thisdict.keys()
print(x)
```

output:: **dict_keys(['brand', 'model', 'year'])**

Dictionaries in Python (cont)

The **values()** method will return a list of all the values in the dictionary.

```
thisdict = { "brand": "Ford", "model": "Mustang", "year": 1964 }
x = thisdict.values()
print(x)
```

output::**dict_values(['Ford', 'Mustang', 1964])**

we can change the value of a specific item by referring to its **key name:**

```
thisdict = { "brand": "Ford", "model": "Mustang", "year": 1964 }
thisdict["year"] = 2018
print(thisdict)
```

output::**{'brand': 'Ford', 'model': 'Mustang', 'year': 2018}**

The **update()** method will update the dictionary with
the items from the given argument.

```
thisdict = { "brand": "Ford", "model": "Mustang", "year": 1964 }
thisdict.update({"year": 2020})
print(thisdict)
```

output::**{'brand': 'Ford', 'model': 'Mustang', 'year': 2020}**

Adding an item to the dictionary is done by using a new index key and assigning a value to it:

```
thisdict = {"brand": "Ford", "model": "Mustang", "year": 1964 }
thisdict["color"] = "red"
print(thisdict)
```

output::
{'brand': 'Ford', 'model': 'Mustang', 'year': 1964, 'color': 'red'}

Dictionaries in Python (cont)

You can loop through a dictionary by using a **for loop**.

```
thisdict = { "brand": "Ford", "model": "Mustang", "year": 1964 }
for x in thisdict:
print(x)
```

output::

brand
model
year

Make a copy of a dictionary with the **copy()** method:

```
thisdict = { "brand": "Ford", "model": "Mustang", "year": 1964 }
mydict = thisdict.copy()
print(mydict)
```

output:: **{'brand': 'Ford', 'model': 'Mustang', 'year': 1964}**

A dictionary can contain dictionaries, this is called **nested dictionaries**.

```
myfamily = {
"child1" : {
"name" : "Emil",
"year" : 2004
},
"child2" : {
"name" : "Tobias",
"year" : 2007
},
"child3" : {
"name" : "Linus",
"year" : 2011
}
}
print(myfamily)
```

output::

{'child1': {'name': 'Emil', 'year': 2004}, 'child2': {'name': 'Tobias', 'year': 2007}, 'child3': {'name': 'Linus', 'year': 2011}}

Dictionaries in Python (cont)

To access items from a nested dictionary, you use the name of the dictionaries, starting with the outer dictionary:

```
myfamily = {
"child1" : {
"name" : "Emil",
"year" : 2004
},
"child2" : {
"name" : "Tobias",
"year" : 2007
},
"child3" : {
"name" : "Linus",
"year" : 2011
}
}
print(myfamily["child2"]["name"])
```

output::**Tobias**

setdefault() Method

```
car = {
"brand": "Ford",
"model": "Mustang",
"year": 1964
}
x = car.setdefault("model", "Bronco")
print(x)
```

output:: **Mustang**

As of Python version 3.7, dictionaries are ordered. In Python 3.6 and earlier, dictionaries are unordered.

Python Strings

Strings

Strings in python are surrounded by either single quotation marks,
or double quotation marks.
Example::

```
print("Hello")
print('Hello')
```

output::

Hello
Hello

Assign String to a Variable

Python Strings (cont)

Example::

```
a = "Hello"
print(a)
```

output:: {nl}} **Hello**

Multiline Strings

Example::

```
a = """Lorem ipsum dolor sit amet,
consectetur adipiscing elit,
sed do eiusmod tempor incididunt {nl}} ut
labore et dolore magna aliqua."""
print(a)
```

output::

Lorem ipsum dolor sit amet,
consectetur adipiscing elit,
sed do eiusmod tempor incididunt
ut labore et dolore magna aliqua.

Strings are Arrays

Example::
Get the character at position 1 (remember that the first character has the position 0):

```
a = "Hello, World!"
print(a[1]) {nl}}output::
e
```

Looping Through a String

Example::

```
for x in "banana":
print(x)
```

output:: {nl}}output::
**b
a
n
a
n
a

String Length

The len() function returns the length of a string:
Example::

```
a = "Hello, World!"
print(len(a))
```

output::

13

Check String

Python Strings (cont)

Example::

txt = "The best things in life are free!"

print("free" in txt)

output:: {nl}}**True**

Use it in an if statement:

Example::

txt = "The best things in life are free!"

if "free" in txt:

print("Yes, 'free' is present.") {nl}}output::

Yes, 'free' is present.

Check if NOT

Example::

txt = "The best things in life are free!"

print("expensive" not in txt)

output:: {nl}}**True**

using 'if'

Example::

txt = "The best things in life are free!"

if "expensive" not in txt:

print("No, 'expensive' is NOT present.")

{nl}}output::

No, 'expensive' is NOT present.

Slicing

Example::

b = "Hello, World!"

print(b[2:5])

output::

llo

Modify Strings to Upper Case

Example::

a = "Hello, World!"

print(a.upper())

output::

HELLO, WORLD!

Modify Strings to Lower Case

Example::

a = "Hello, World!"

print(a.lower())

output::

hello, world!

Remove Whitespace

Python Strings (cont)

The strip() method removes any whitespace from the beginning or the end:

Example::

a = " Hello, World! "

print(a.strip())

output::

Hello, World!

Replace String

Example::

a = "Hello, World!"

print(a.replace("H", "J"))

output::

Jello, World!

Split String

Example::

a = "Hello, World!"

b = a.split(",")

print(b)

output::

['Hello', ' World!']

String Concatenation

Example::

a = "Hello"

b = "World"

c = a + b

print(c)

output::

HelloWorld

To add a space between them, add a " ":

Example::

a = "Hello"

b = "World"

c = a + " " + b

print(c)

output::

Hello World

Format - Strings

Python Strings (cont)

Use the format() method to insert numbers into strings:

Example::

age = 30

txt = "My name is Srinivas, and I am {}"

print(txt.format(age))

output::

My name is Srinivas, and I am 30

You can use index numbers {0} to be sure the

arguments are placed in the correct placeh-olders:

Example::

quantity = 3

itemno = 567

price = 49.95

myorder = "I want to pay {2} dollars for {0} pieces of item {1}."

print(myorder.format(quantity, itemno, price))

output::

I want to pay 49.95 dollars for 3 pieces of item 567

Escape Characters

The escape character allows you to use double quotes

when you normally would not be allowed:

Example::

txt = "Iam \"Important\" in this line."

print(txt)

output::

Iam "Important" in this line

Capitalize() Method In Strings

The first character is converted to upper case, and

the rest are converted to lower case:

Example::

txt = "python is FUN!"

x = txt.capitalize()

print (x)

output::

Python is fun!

String count() Method

Python Strings (cont)

Return the number of times the value "apple" appears in the string:
Example::
txt = "I love apples, apple are my favorite fruit"
x = txt.count("apple")
print(x)
output::
2

String endswith() Method

Example::
txt = "Hello, welcome to my world."
x = txt.endswith(".")
print(x)
output::
True

String find() Method

Example::
txt = "Hello, welcome to my world."
x = txt.find("welcome")
print(x)
output::
7

String isalnum() Method

Example::
txt = "Company12"
x = txt.isalnum()
print(x)
output::
True

String isalpha() Method

Example::
txt = "CompanyX"
x = txt.isalpha()
print(x)
output::
True

Strings in python are surrounded by either single quotation marks, or double quotation marks.

Tuples In Python

Tuple:

Tuples are used to store multiple items in a single variable.
A tuple is a collection which is ordered and unchangeable.
Tuples are written with round brackets.
Example::
thistuple = ("apple", "banana", "cherry")
print(thistuple).
output::**('apple', 'banana', 'cherry')**

Tuple Items

Tuple items are ordered, unchangeable, and allow duplicate values.
Tuple items are indexed, the first item has index [0],
the second item has index [1] etc.

Ordered

When we say that tuples are ordered,
it means that the items have a defined order,
and that order will not change.

Unchangeable

Tuples are unchangeable, meaning that we cannot change,
add or remove items after the tuple has been created.

Allow Duplicates

Example::
thistuple = ("apple", "banana", "cherry", "apple", "cherry")
print(thistuple)
output::
('apple', 'banana', 'cherry', 'apple', 'cherry')

Tuples In Python (cont)

Create Tuple With One Item

To create a tuple with only one item,
you have to add a comma after the item,
otherwise Python will not recognize it as a tuple.
thistuple = ("apple",)
print(type(thistuple))
#NOT a tuple
thistuple = ("apple")
print(type(thistuple))
output::
<class 'tuple'>
<class 'str'>

Access Tuple Items

Print the second item in the tuple:
thistuple = ("apple", "banana", "cherry")
print(thistuple[1])
output::
banana

Change Tuple Values

Once a tuple is created, you cannot change its values as they are called immutable.
But there is a workaround. You can convert the tuple into a list,
change the list, and convert the list back into a tuple.
x = ("apple", "banana", "cherry")
y = list(x)
y[1] = "kiwi"
x = tuple(y)
print(x)
output::
("apple", "kiwi", "cherry")

Add tuple to a tuple.

Tuples In Python (cont)

Example::
Create a new tuple with the value "orange", and add that tuple:

```
thistuple = ("apple", "banana", "cherry")
y = ("orange",)
thistuple += y
print(thistuple)
```

output::

```
('apple', 'banana', 'cherry', 'orange')
```

Unpacking a Tuple

When we create a tuple, we normally assign values to it.
This is called "packing" a tuple:
Packing a tuple:

```
fruits = ("apple", "banana", "cherry")
print(fruits)
```

output::

```
('apple', 'banana', 'cherry')
```

But, in Python, we are also allowed to extract the values back into variables.
This is called "unpacking":
unpack Tuple

```
fruits = ("apple", "banana", "cherry")
(green, yellow, red) = fruits
print(green)
print(yellow)
print(red)
```

output::

```
apple
banana
cherry
```

Using Asterisk*

Tuples In Python (cont)

If the number of variables is less than the number of values,
you can add an * to the variable name and the values
will be assigned to the variable as a list:

Example::

```
fruits = ("apple", "banana", "cherry", "strawberry", "raspberry")
(green, yellow, *red) = fruits
print(green)
print(yellow)
print(red)
```

output::

```
apple
banana
['cherry', 'strawberry', 'raspberry']
```

Loop Through a Tuple

You can loop through the tuple items by using a for loop.
Example::

```
thistuple = ("apple", "banana", "cherry")
for x in thistuple:
print(x)
```

output::

```
apple
banana
cherry
```

Loop Through the Index Numbers In Tuples

Use the range() and len() functions to create a suitable iterable.
Example::

```
thistuple = ("apple", "banana", "cherry")
for i in range(len(thistuple)):
print(thistuple[i])
```

output::

```
apple
banana
cherry
```

Using a While Loop In Tuples

Tuples In Python (cont)

Example::

```
thistuple = ("apple", "banana", "cherry")
i = 0
while i < len(thistuple):
print(thistuple[i])
i = i + 1
```

output::

```
apple
banana
cherry
```

Join Two Tuples

To join two or more tuples you can use the + operator:
Example::

```
tuple1 = ("a", "b" , "c")
tuple2 = (1, 2, 3)
tuple3 = tuple1 + tuple2
print(tuple3)
```

output::

```
('a', 'b', 'c', 1, 2, 3)
```

Multiply Tuples

Example::

```
fruits = ("apple", "banana", "cherry")
mytuple = fruits * 2
print(mytuple)
```

output::

```
('apple', 'banana', 'cherry', 'apple', 'banana', 'cherry')
```

Tuple count() Method

Example::

```
thistuple = (1, 3, 7, 8, 7, 5, 4, 6, 8, 5)
x = thistuple.count(5)
print(x)
```

output::

```
2
```

Tuple index() Method

Tuples In Python (cont)

Example::

```
thistuple = (1, 3, 7, 8, 7, 5, 4, 6, 8, 5)
x = thistuple.index(8)
print(x)
```

output::

```
3
```

When creating a tuple with only one item, remember to include a comma after the item, otherwise it will not be identified as a tuple.

By **srinivas.ram**
cheatography.com/srinivas-ram/

Not published yet.
Last updated 8th April, 2023.