THEORY QUESTIONS ASSIGNMENT

1. Python theory questions

1. What is Python and what are its main features?

Python is a programming language, he main features are : high-level, general-purpose, easy to code, easy to read, free and open-source, Large Standard Library

2. Discuss the difference between Python 2 and Python 3

Python 3 is a newer version of the Python programming language , This version was mainly released to fix problems that exist in **Python 2**

Key differences:

- Python 3 syntax is simpler and easily understandable whereas Python 2 syntax is comparatively difficult to understand.
- Python 3 default storing of strings is Unicode whereas Python 2 stores need to define Unicode string value with "u."
- Python 3 value of variables never changes whereas in Python 2 value of the global variable will be changed while using it inside for-loop.
- Python 3 exceptions should be enclosed in parenthesis while Python 2 exceptions should be enclosed in notations.
- Python 3 rules of ordering comparisons are simplified whereas Python 2 rules of ordering comparison are complex.
- Python 3 offers Range() function to perform iterations whereas, In Python 2, the xrange() is used for iterations.

3. What is PEP 8?

Style Guide for Python Code

4. In computing / computer science what is a program?

A program is a set of instructions that a computer follows in order to perform a particular task

5. In computing / computer science what is a process?

It is an executing program in computer memory.

6. In computing / computer science what is cache?

A cache is a block of memory for storing data which is likely used again

7. In computing / computer science what is a thread and what do we mean by

multithreading?

A thread is an execution context, which is all the information a CPU needs to execute a stream of instructions

Multithreading enables us to run multiple threads concurrently.

8. In computing / computer science what is concurrency and parallelism and what

are the differences?

concurrency is when multiple sequences of operations are run in overlapping periods of time,

Parallelism refers to techniques to make programs faster by performing several computations at the same time

9. What is GIL in Python and how does it work?

global interpreter lock: An interpreter that uses GIL always allows exactly one thread to execute at a time

10. What do these software development principles mean: DRY, KISS, BDUF

Dry :Don't Repeat Yourself, When writing your code, don't repeat yourself

KISS : Keep It Simple Stupid, ensure your programming is simple and clear to understand

BDUF: Big Design Up Front, developer should complete the project's design first. After that, they can now implement it.

11. What is a Garbage Collector in Python and how does it work?

Garbage collection is to release memory when the object is no longer in use

It works through those two ways : Reference counting/Generational garbage collection

12. How is memory managed in Python?

The Python memory is primarily managed by Python private heap space.

13. What is a Python module?

A python file containing Python code, containing Python statements and definitions

14. What is docstring in Python?

A Python docstring is a string used to document a Python module, class, function or method, so programmers can understand what it does without having to read the details of the implementation.

15. What is pickling and unpickling in Python? Example usage.

Pickling is the process by which Python objects are converted to byte streams

```
import pickle

data = [ { 'a':'A', 'b':2, 'c':3.0 } ]

example = pickle.dumps(data)
```

Unpickling is the process of converting a byte stream into the python object.

```
retreived_dict = pickle.load(example)
```

16. What are the tools that help to find bugs or perform static analysis?

Pychecker and Pylint

17. How are arguments passed in Python by value or by reference?

Answer: Both of them

18. What are Dictionary and List comprehensions in Python? Provide examples.

Dictionary: data structure, consists of a collection of key-value pairs. Each key-value pair maps the key to its associated value

```
fruit = {'mango':40, 'banana':10, 'cherry':20}
```

List comprehensions: an way to define and create lists based on existing lists whilst altering or filtering elements

```
fruits = ["apple", "banana", "cherry", "kiwi", "mango"]

newlist = [x for x in fruits if "a" in x]
```

19. What is namespace in Python?

A namespace is a system that has a unique name for each and every object in Python

20.What is pass in Python?

nothing happens when the pass is executed, it is used as a placeholder for future code

21. What is unit test in Python?

Unit testing makes code future proof since you anticipate the cases where your code could potentially fail or produce a bug, python has package called unittest

22. In Python what is slicing?

You can get certain characters in a sequence by slicing,

23. What is a negative index in Python?

In python The negative indexing starts from where the array ends.

24.How can the ternary operators be used in python? Give an example.

It testing a condition in a single line

```
a, b = 1, 2

 min = a if a < b else b
```

25. What does this mean: *args, **kwargs? And why would we use it?

Both of them allow passing multiple arguments or keyword arguments to a function

*args: it allows passing a varying number of positional arguments

**kwargs : basically same, apart from this only accepts keyword arguments

26. How are range and xrange different from one another?

Range : returns a range object

Xrange: returns the generator object that can be used to display numbers only by looping.

27. What is Flask and what can we use it for?

Flask is web framework, use it for :

Integrated support for unit testing

Easily deployable in production

Easy to create APIs

Supports Visual Debugging

Can be easily integrated with all the major databases

28. What are clustered and non-clustered index in a relational database?

Clustered index is primary key only one column, non-clustered index can be multiple columns,

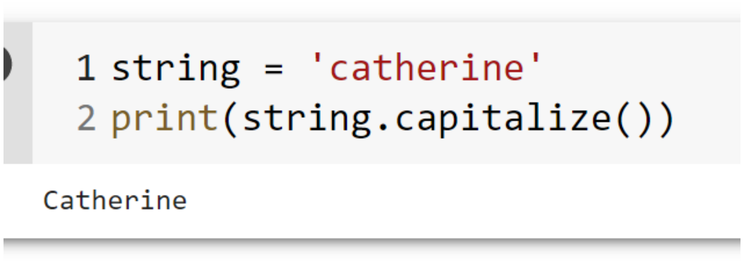
29. What is a 'deadlock' a relational database?

Deadlock is when two concurrent transactions cannot make progress because each one waits for the other to release a lock

30.What is a 'livelock' a relational database?

Livelock is a deadlock-like situation in which processes block each other with a repeated state change yet make no progress

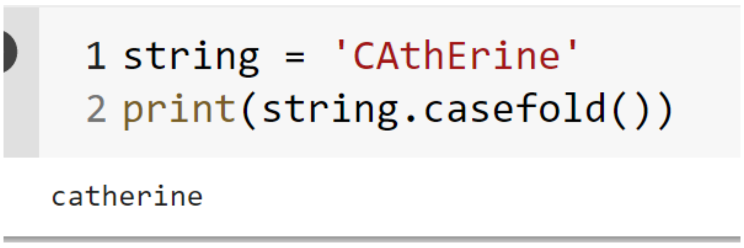
capitalize() : make the first character in a string uppercase,



```
string = 'catherine'
print(string.capitalize())
```

```
Catherine
```

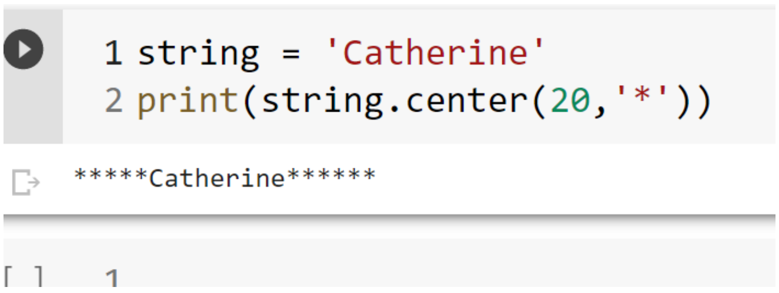
casefold(): It makes everything character of a string lowercase



```python
string = 'CAthErine'
print(string.casefold())
```

```
catherine
```

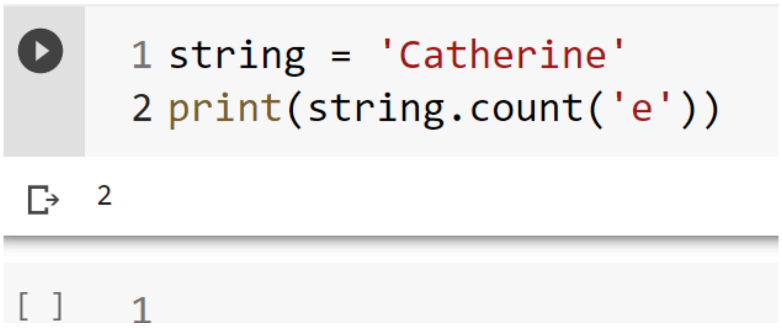
center(): center align the string, using a specified character (space is default) as the fill character.



```python
string = 'Catherine'
print(string.center(20,'*'))
```

```
*****Catherine******
```

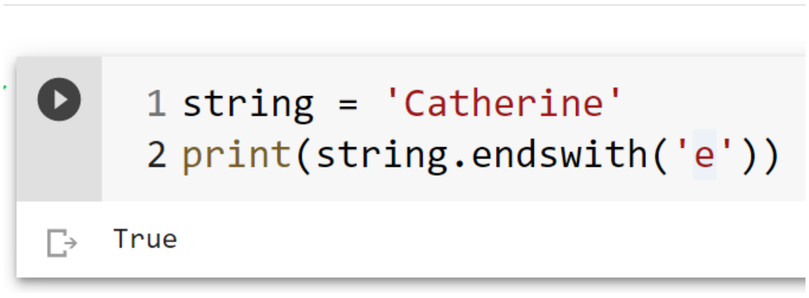
count(): returns the number of elements with the specified value.



```python
string = 'Catherine'
print(string.count('e'))
```

```
2
```

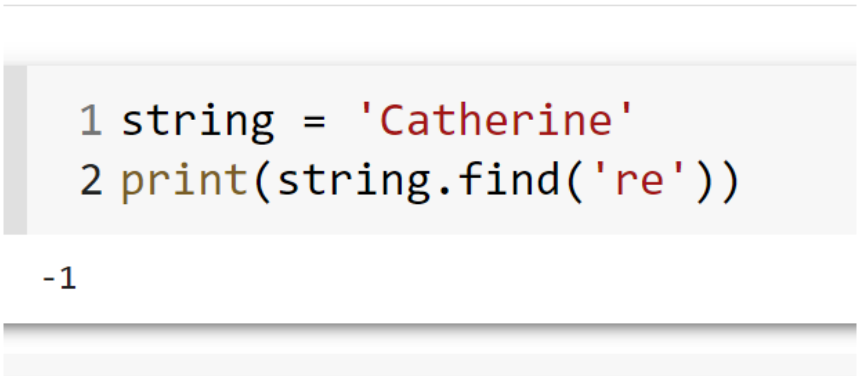
endswith(): returns True if the string ends with the specified value, otherwise False.



```python
string = 'Catherine'
print(string.endswith('e'))
```

```
True
```

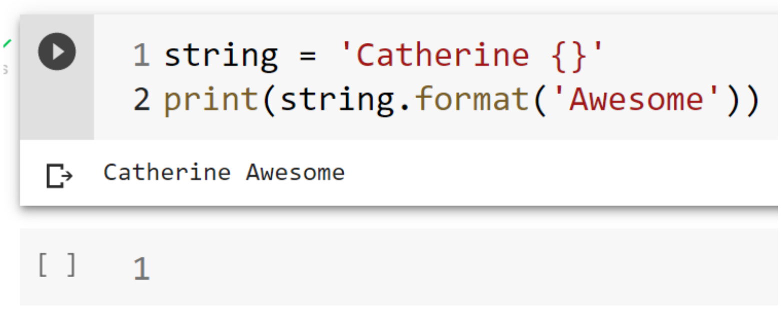
find(): finds the index of the specified value, return -1 if not found



```python
string = 'Catherine'
print(string.find('re'))
```

```
-1
```

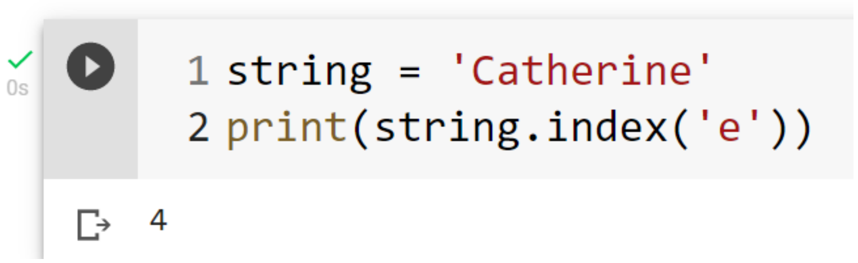
format(): takes unlimited number of arguments, and are placed into the respective placeholders



```
string = 'Catherine {}'
print(string.format('Awesome'))
```

```
Catherine Awesome
```

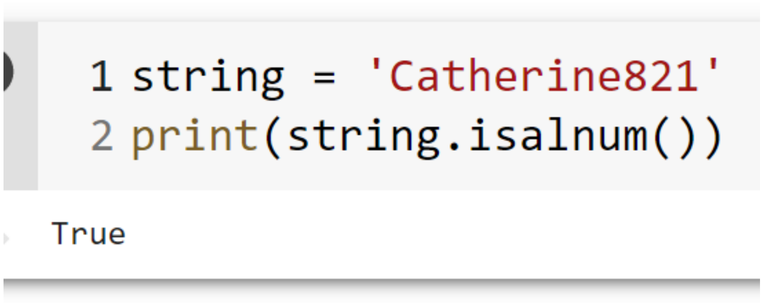
index(): returns the position at the first occurrence of the specified value in a string



```
string = 'Catherine'
print(string.index('e'))
```

```
4
```

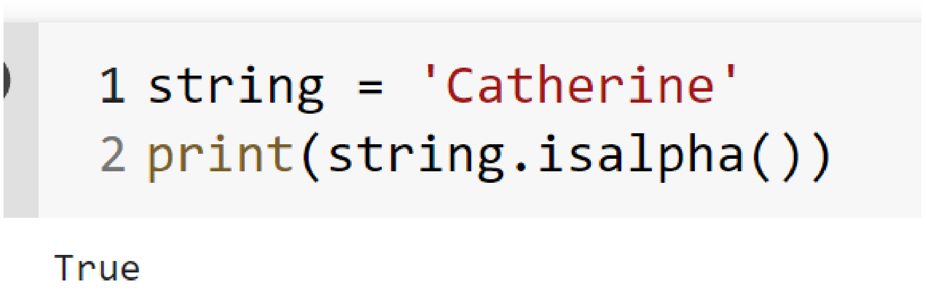
isalnum():returns True if all the characters are alphanumeric, otherwise return False



```
string = 'Catherine821'
print(string.isalnum())
```

```
True
```

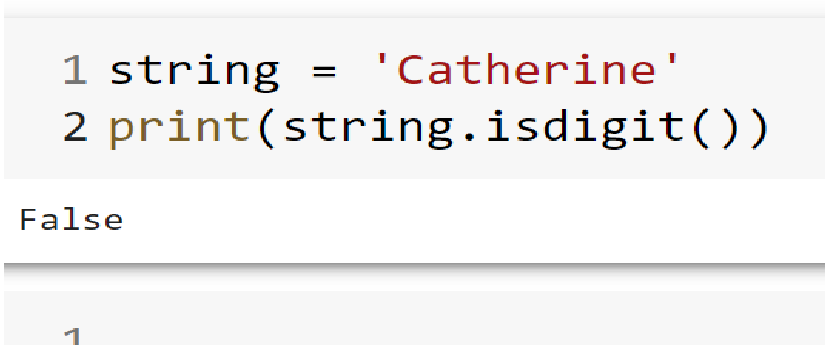
isalpha(): returns True if all the characters are alphas, otherwise return False



```
string = 'Catherine'
print(string.isalpha())
```

```
True
```

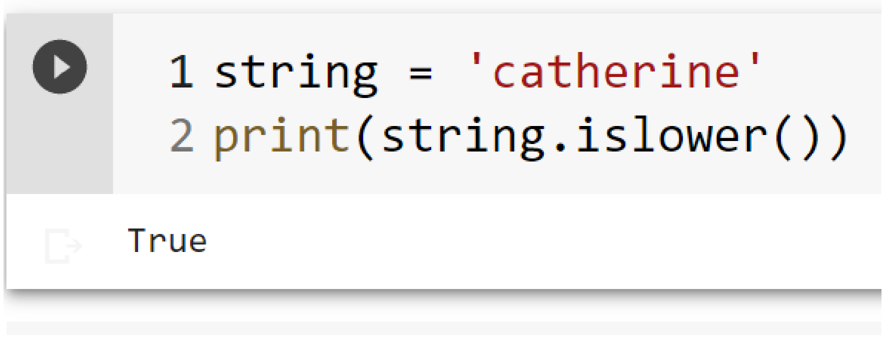
isdigit(): returns True if all characters are numeric, otherwise return False



```python
string = 'Catherine'
print(string.isdigit())
```

```
False
```

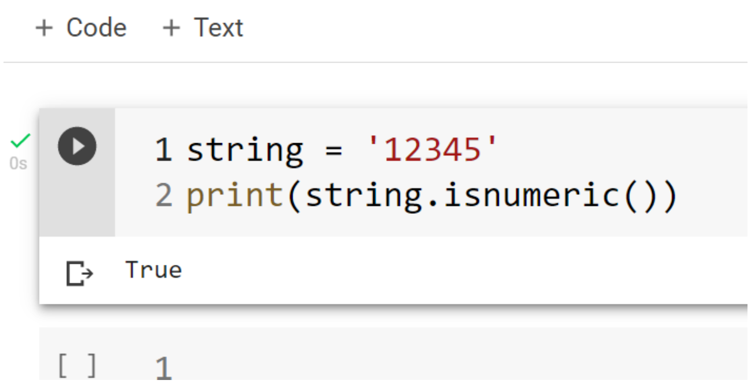
islower(): returns True if all characters are in Lowercase, otherwise return False



```python
string = 'catherine'
print(string.islower())
```

```
True
```

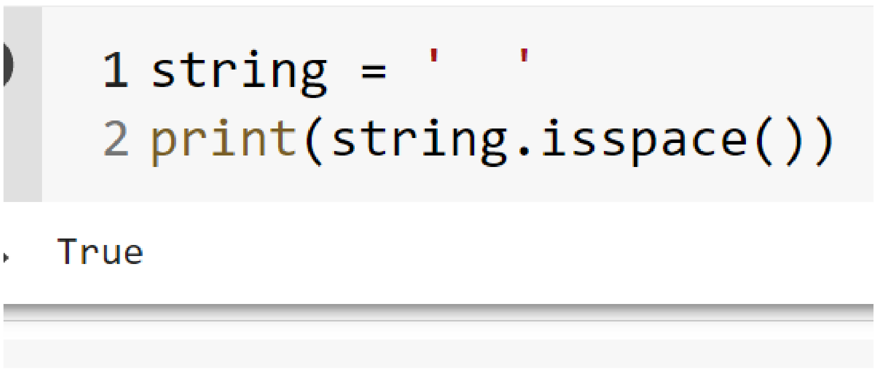
isnumeric(): returns True if all the characters are numeric (0-9), otherwise False.



```python
string = '12345'
print(string.isnumeric())
```

```
True
```

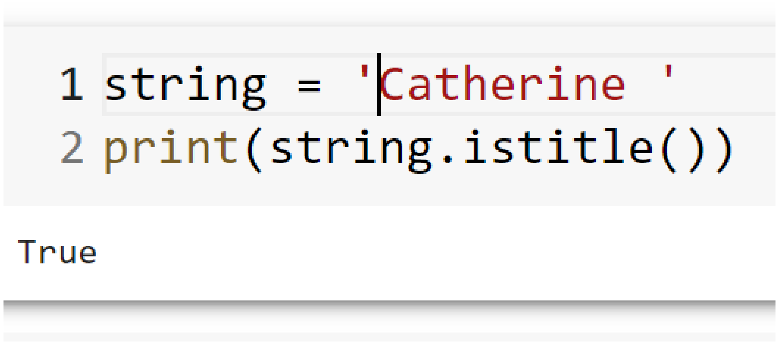
isspace():returns True if all the characters in a string are whitespaces, otherwise return False



```python
string = '   '
print(string.isspace())
```

```
True
```

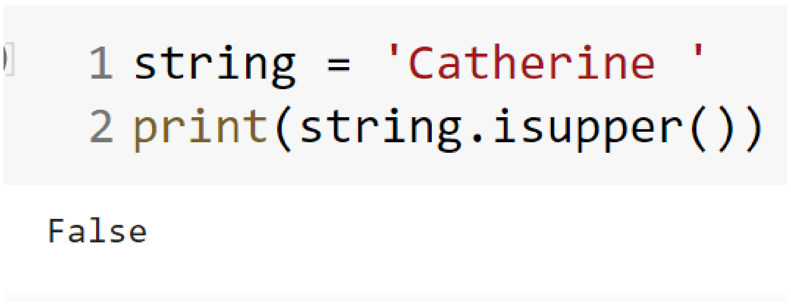
istitle(): returns True if all words in a text start with a upper case letter, and the rest of the word are lower case letters, otherwise False.



```python
string = 'Catherine '
print(string.istitle())
```

```
True
```

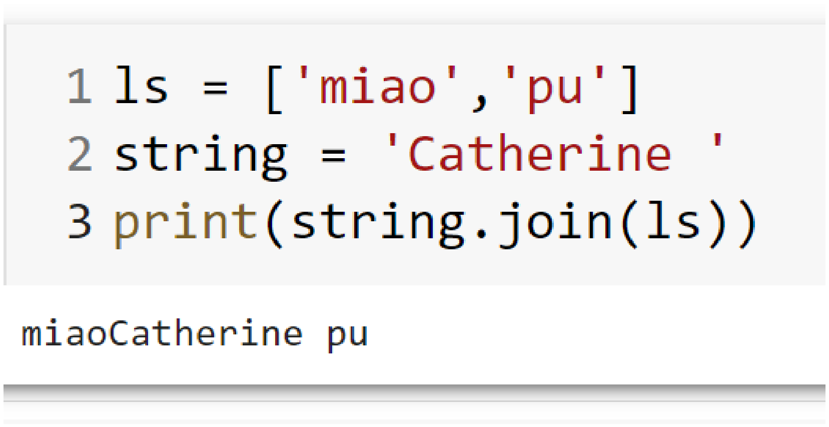
isupper() (): returns True if all characters are in uppercase letters, otherwise return False



```python
string = 'Catherine '
print(string.isupper())
```

```
False
```

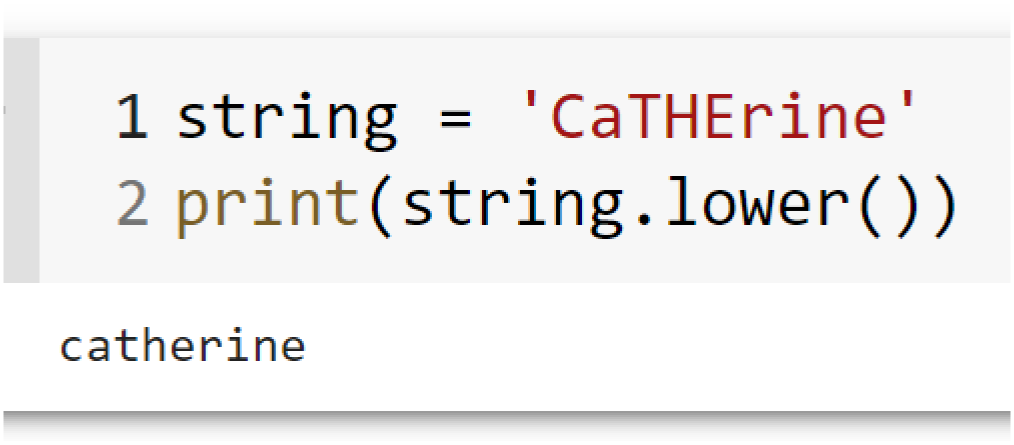
join():takes all items in an iterable and joins them into one string



```python
ls = ['miao','pu']
string = 'Catherine '
print(string.join(ls))
```

```
miaoCatherine pu
```

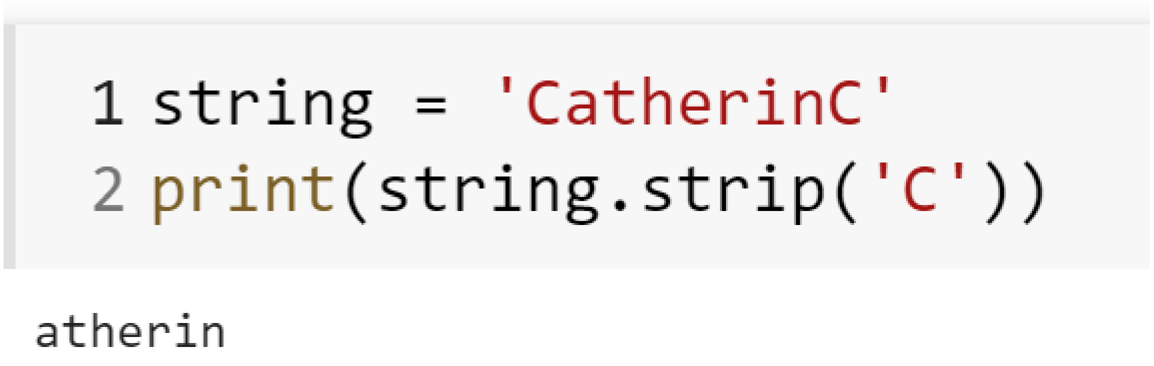
lower():returns a string where all characters are lower case



```python
string = 'CaTHErine'
print(string.lower())
```

```
catherine
```

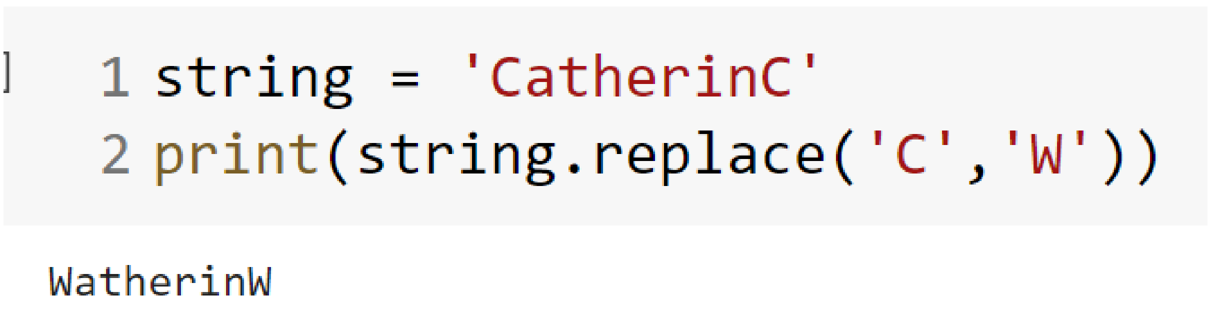
strip(): remove the characters you specify at the beginning and end



```python
string = 'CatherinC'
print(string.strip('C'))
```

```
atherin
```

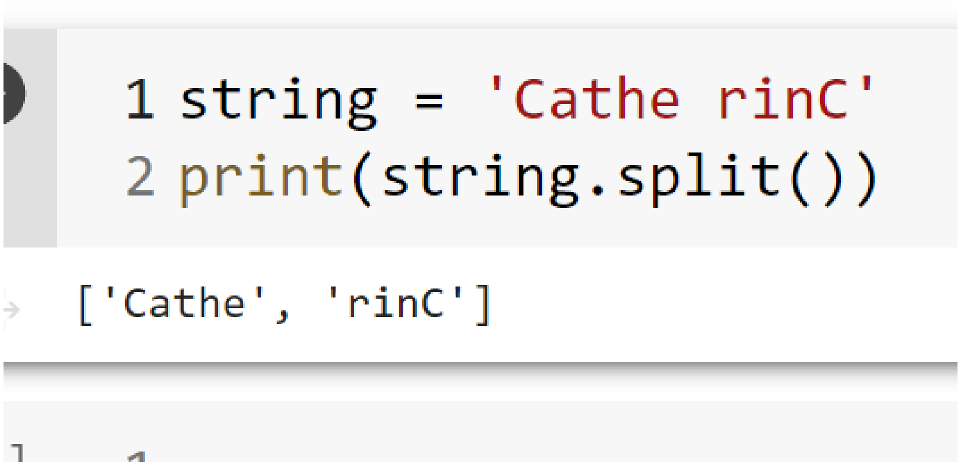
replace(): remove curtain characters update them with the ones you want to change into



```python
string = 'CatherinC'
print(string.replace('C','W'))
```

```
WatherinW
```

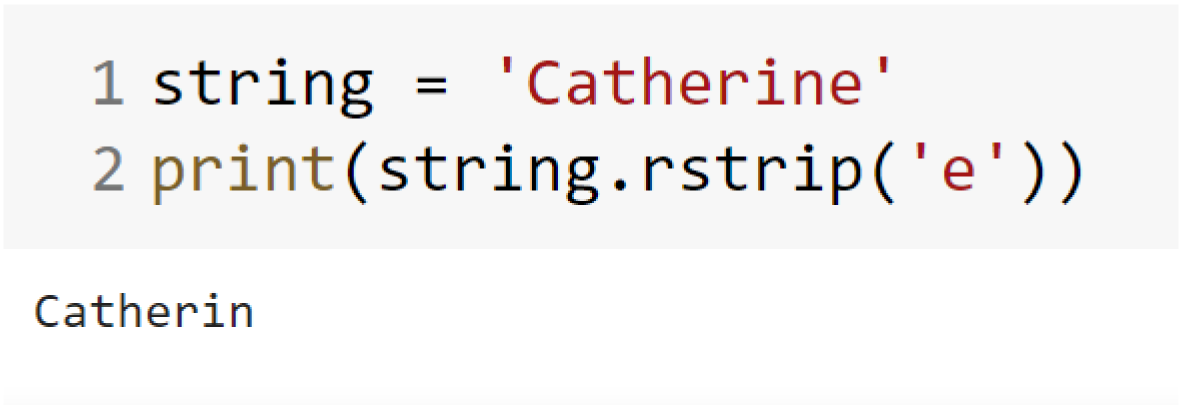
split():splits a string into a list. Default sperate is whitespace



```python
string = 'Cathe rinC'
print(string.split())
```

```
['Cathe', 'rinC']
```

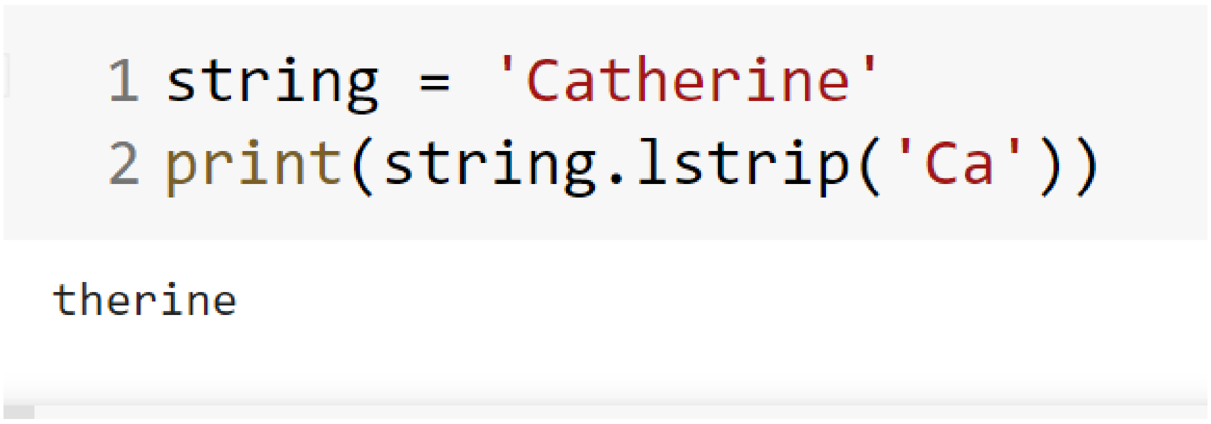
rstrip(): remove characters at the end of a string



```python
string = 'Catherine'
print(string.rstrip('e'))
```

```
Catherin
```

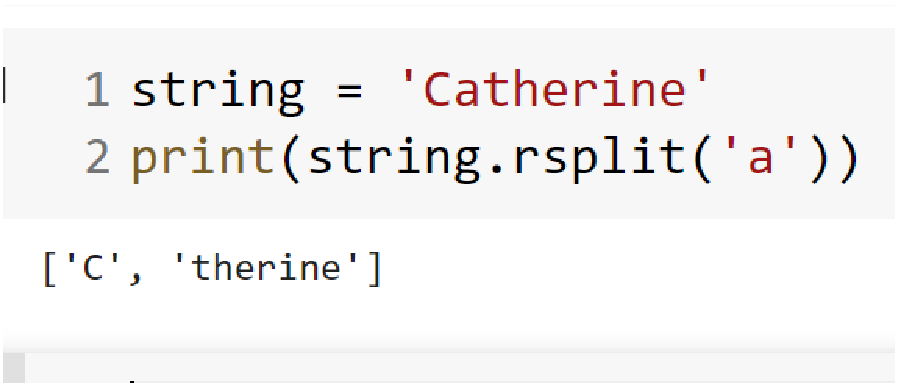
lstrip(): delete all the characters at the beginning of a string



```python
string = 'Catherine'
print(string.lstrip('Ca'))
```

```
therine
```

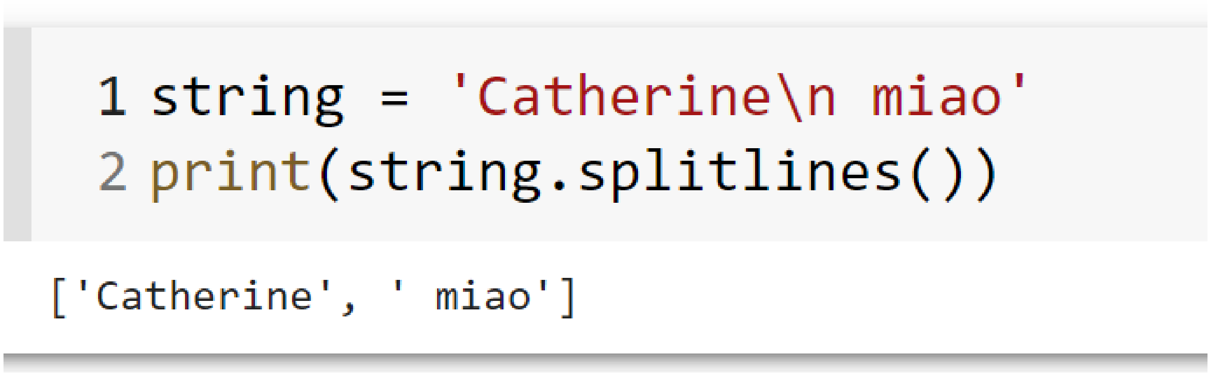
rsplit():splits a string into a list, starting from the right



```python
string = 'Catherine'
print(string.rsplit('a'))
```

```
['C', 'therine']
```

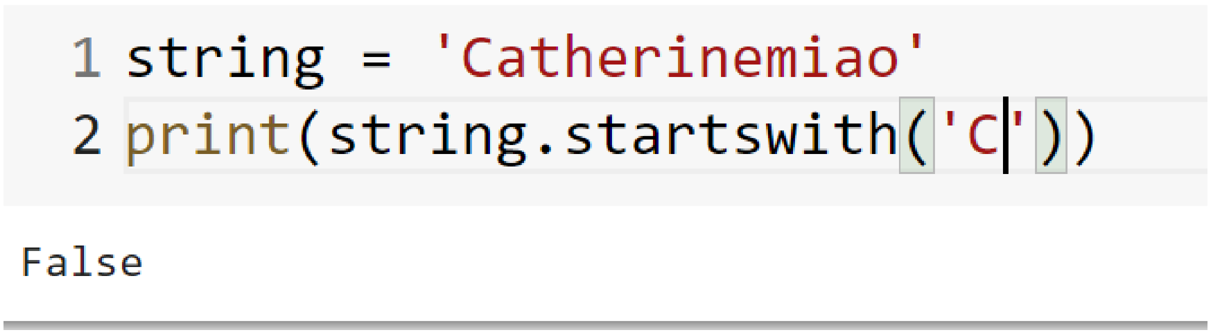
splitlines():splits a string into a list. The splitting is done at line breaks.



```python
string = 'Catherine\n miao'
print(string.splitlines())
```

```
['Catherine', ' miao']
```

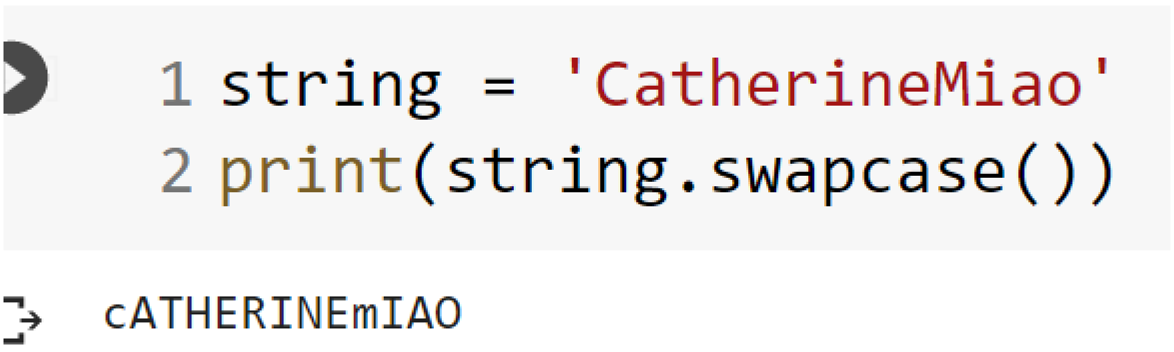
startswith(): returns True if a string starts with the specified value, otherwise return False



```python
string = 'Catherinemiao'
print(string.startswith('C'))
```

```
False
```

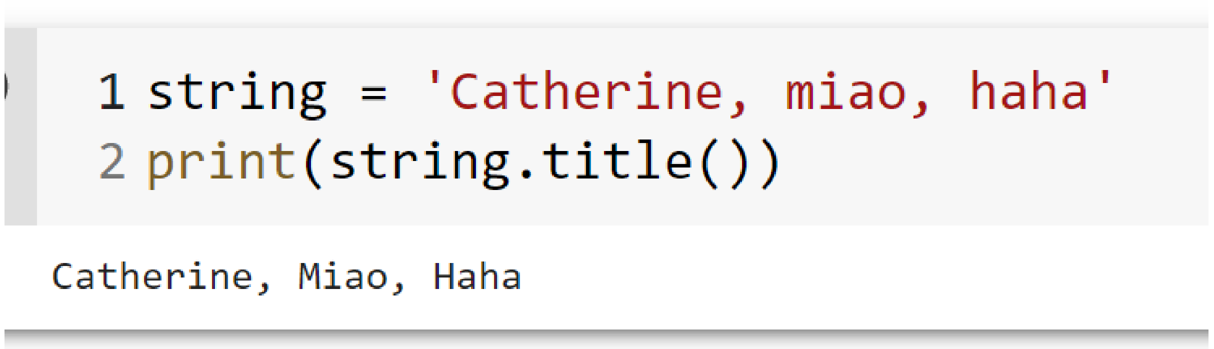
swapcase(): upper case letters become lower case letter, lower become upper



```python
string = 'CatherineMiao'
print(string.swapcase())
```

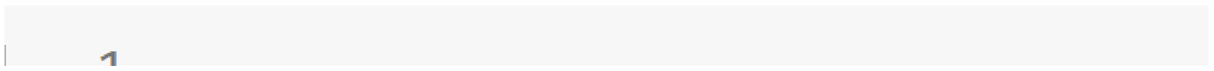
```
cATHERINEmIAO
```

title():returns a string where the first character in every word is upper case

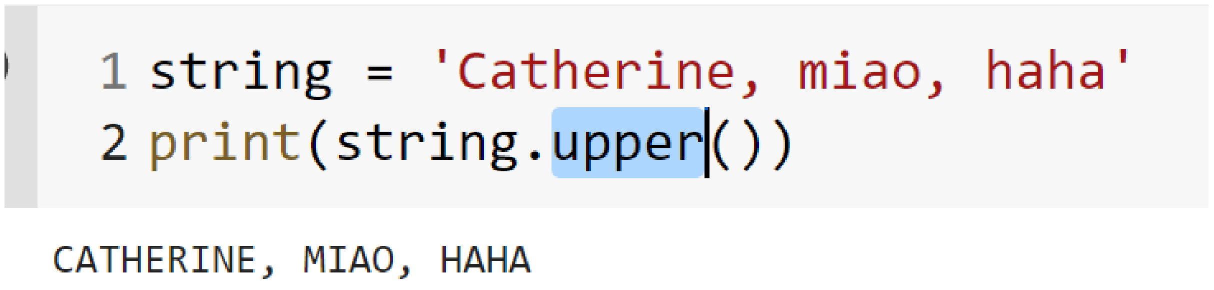


```python
string = 'Catherine, miao, haha'
print(string.title())
```

```
Catherine, Miao, Haha
```



upper():returns a string where all characters are in upper case

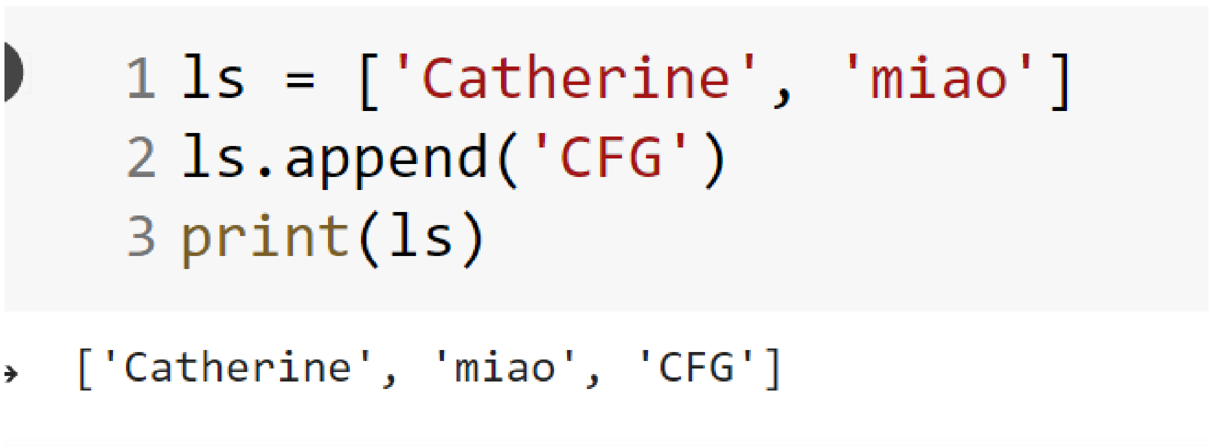


```python
string = 'Catherine, miao, haha'
print(string.upper())
```

```
CATHERINE, MIAO, HAHA
```

List methods

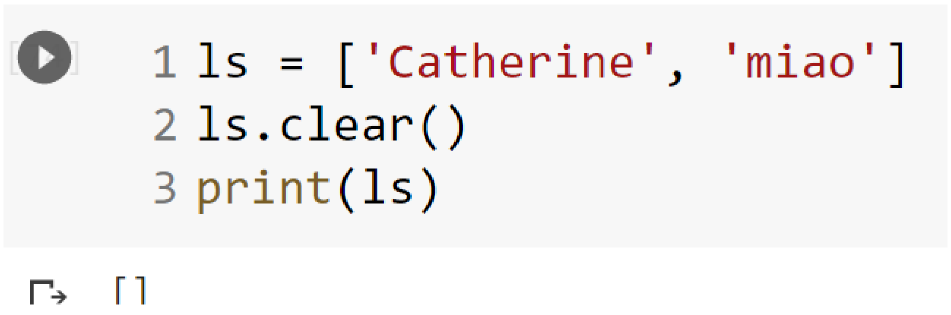
append(): inserting new elements into the end of a list



```python
ls = ['Catherine', 'miao']
ls.append('CFG')
print(ls)
```

```
['Catherine', 'miao', 'CFG']
```

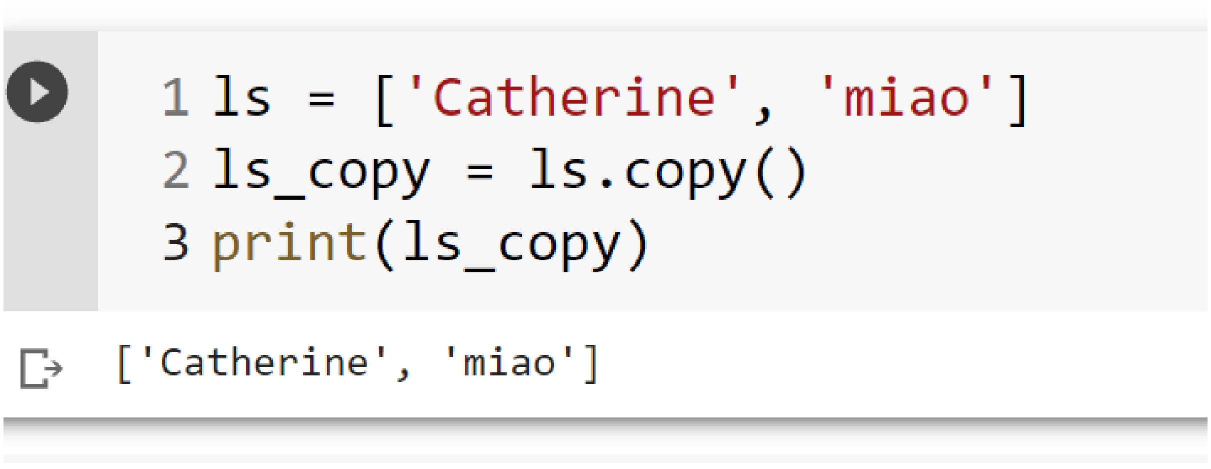
clear(): remove everything from a list



```python
ls = ['Catherine', 'miao']
ls.clear()
print(ls)
```

```
[]
```

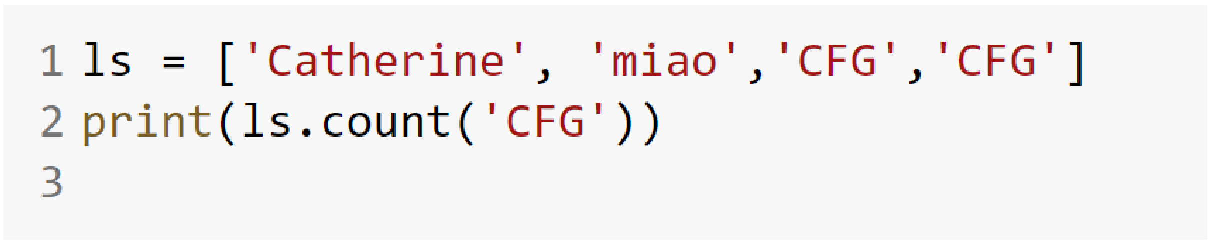
copy(): make a copy of a list



```python
ls = ['Catherine', 'miao']
ls_copy = ls.copy()
print(ls_copy)
```

```
['Catherine', 'miao']
```

count(): count the frequency of a specified value in a list

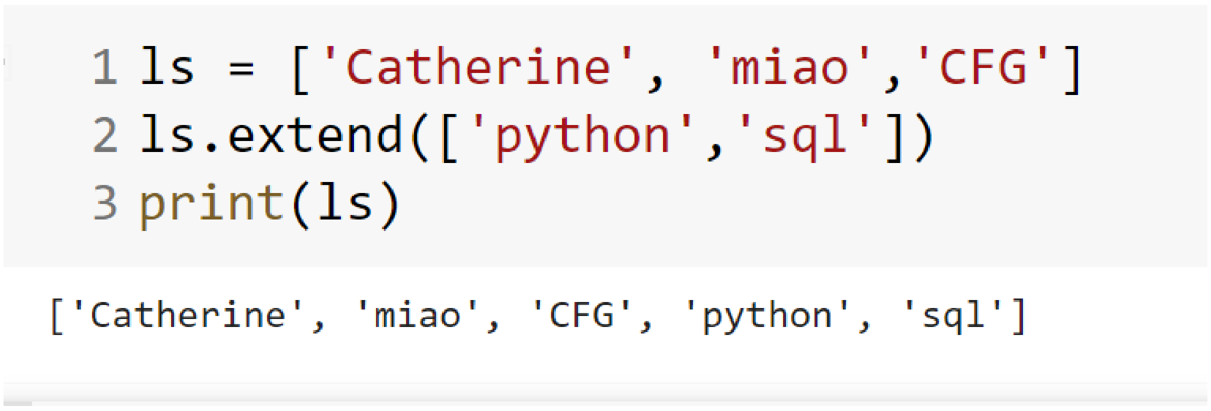


```python
ls = ['Catherine', 'miao','CFG','CFG']
print(ls.count('CFG'))

```

2

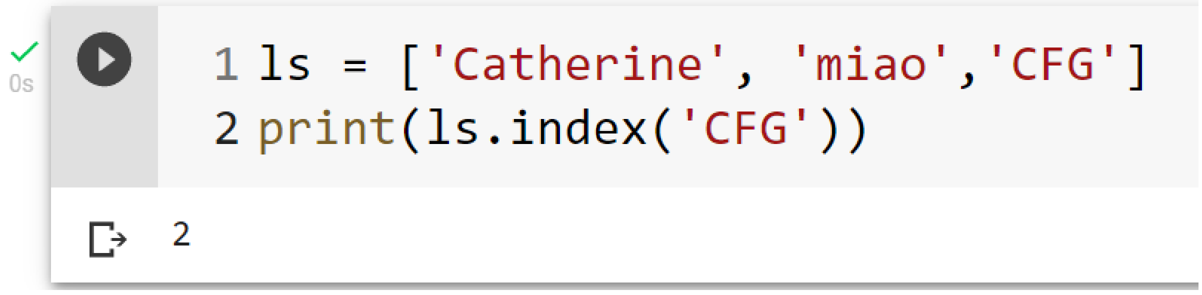
extend(): adds elements in a new list into the end of a list



```python
ls = ['Catherine', 'miao','CFG']
ls.extend(['python','sql'])
print(ls)
```

```
['Catherine', 'miao', 'CFG', 'python', 'sql']
```

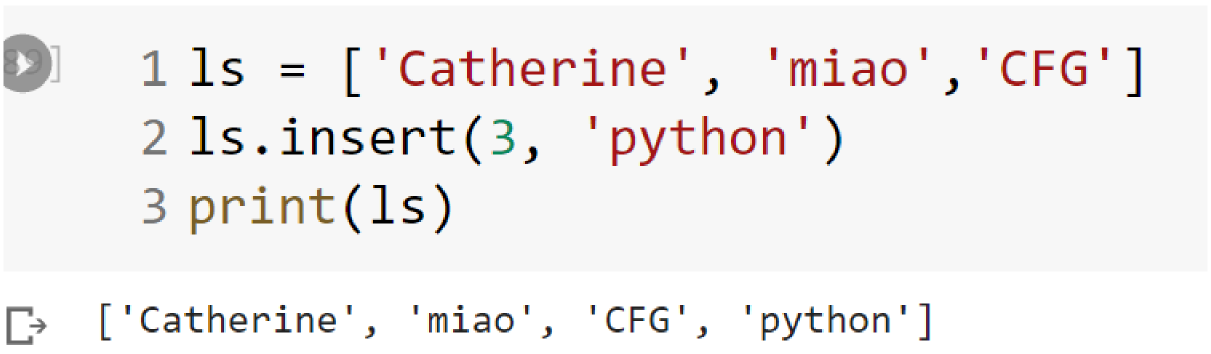
index():returns the position at the first occurrence of the specified value in a list



```python
ls = ['Catherine', 'miao','CFG']
print(ls.index('CFG'))
```

```
2
```

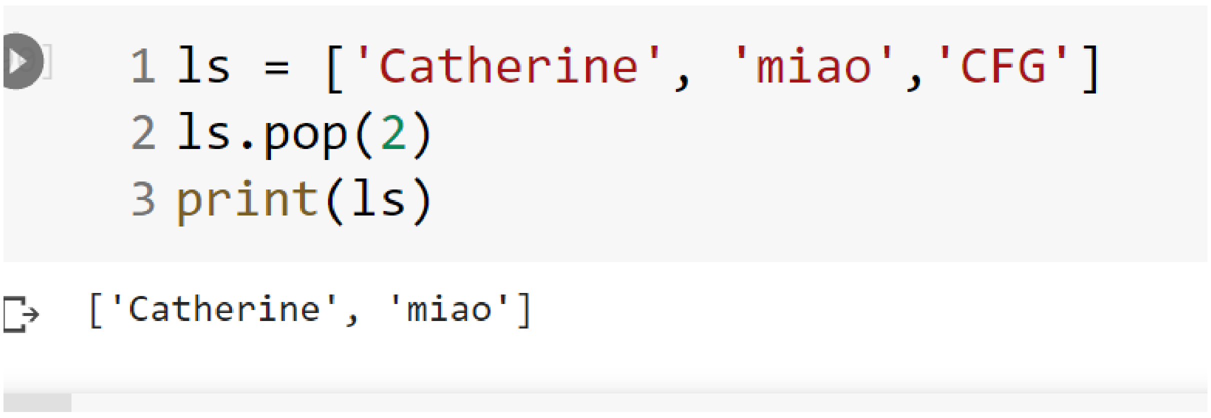
insert(): inserts the specified value at the specified position



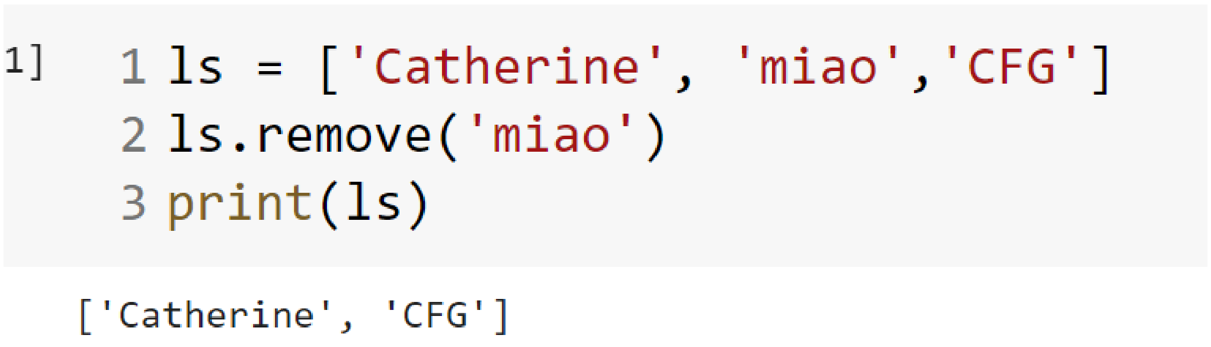
```python
ls = ['Catherine', 'miao','CFG']
ls.insert(3, 'python')
print(ls)
```

```
['Catherine', 'miao', 'CFG', 'python']
```

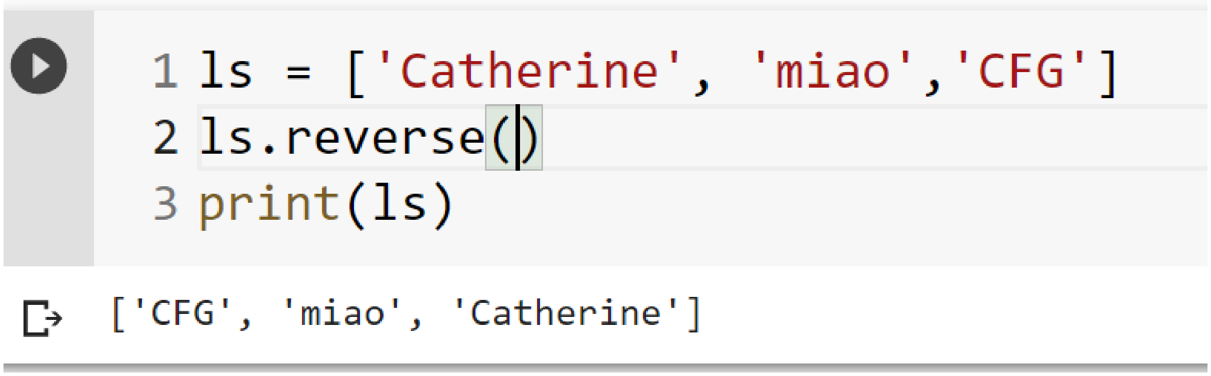
pop():removes the element at the specified position


```python
ls = ['Catherine', 'miao','CFG']
ls.pop(2)
print(ls)
```
```
['Catherine', 'miao']
```

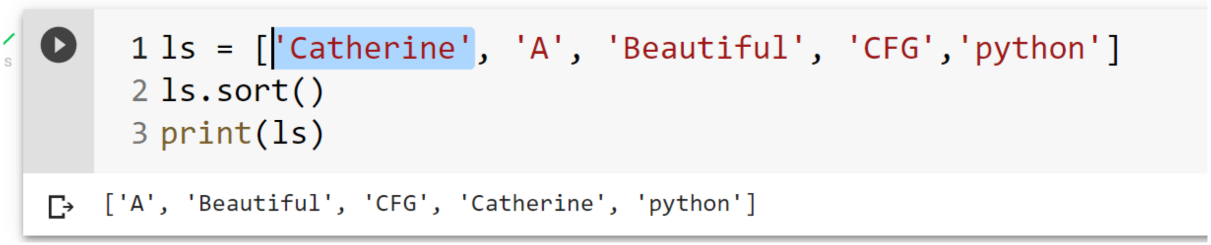
remove():removes the first occurrence of the element with the specified value


```python
ls = ['Catherine', 'miao','CFG']
ls.remove('miao')
print(ls)
```
```
['Catherine', 'CFG']
```

reverse(): reverse a list, the last element becomes the first element

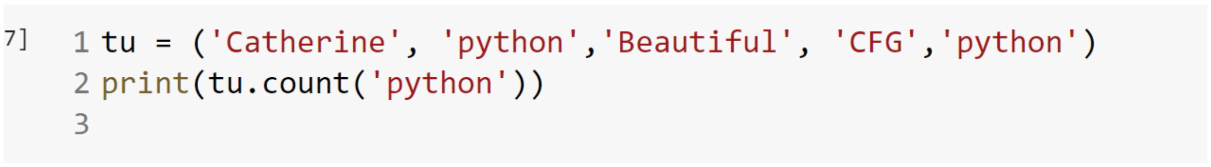

```python
ls = ['Catherine', 'miao','CFG']
ls.reverse()
print(ls)
```
```
['CFG', 'miao', 'Catherine']
```

sort(): sort elements in curtain order in a list , ascending by default


```python
ls = ['Catherine', 'A', 'Beautiful', 'CFG','python']
ls.sort()
print(ls)
```
```
['A', 'Beautiful', 'CFG', 'Catherine', 'python']
```

Python tuple methods:

count():count the frequency of a specified value in a tuple

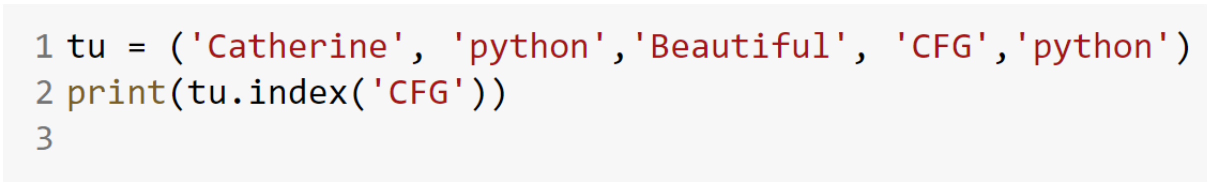


```python
tu = ('Catherine', 'python','Beautiful', 'CFG','python')
print(tu.count('python'))

```

```
2
```

index():returns the position at the first occurrence of the specified value in a tuple



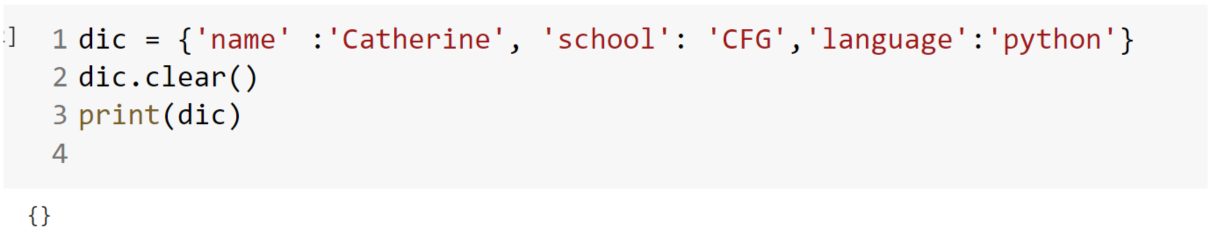
```python
tu = ('Catherine', 'python','Beautiful', 'CFG','python')
print(tu.index('CFG'))

```

```
3
```

Python dictionary methods:

clear(): clear everything in a dict

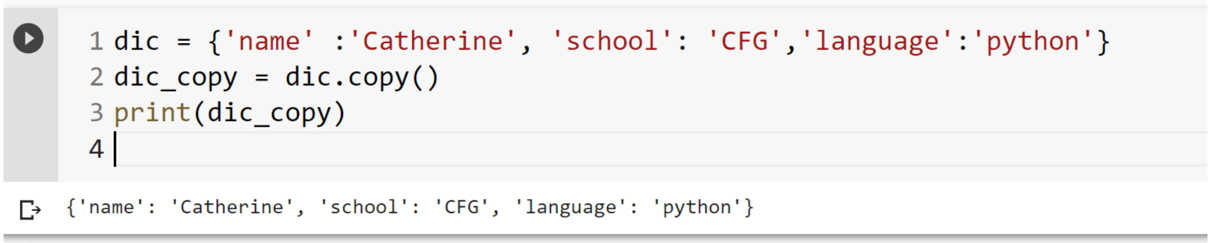


```python
dic = {'name' :'Catherine', 'school': 'CFG','language':'python'}
dic.clear()
print(dic)

```

```
{}
```

copy(): make a copy of a dict

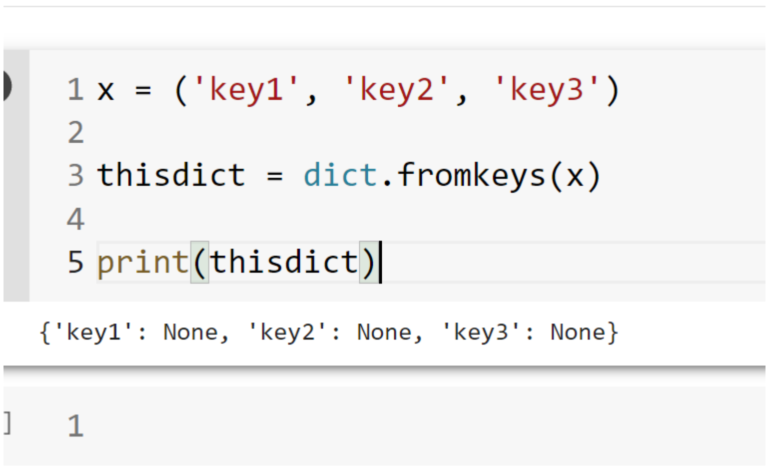


```python
dic = {'name' :'Catherine', 'school': 'CFG','language':'python'}
dic_copy = dic.copy()
print(dic_copy)

```

```
{'name': 'Catherine', 'school': 'CFG', 'language': 'python'}
```

fromkeys(): returns a dict with the specified keys and the specified value



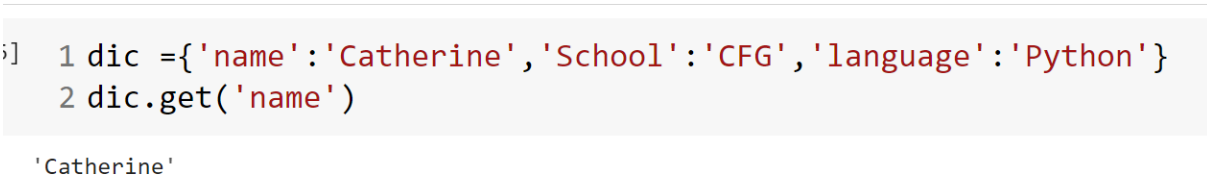
```python
x = ('key1', 'key2', 'key3')

thisdict = dict.fromkeys(x)

print(thisdict)
```

```
{'key1': None, 'key2': None, 'key3': None}
```

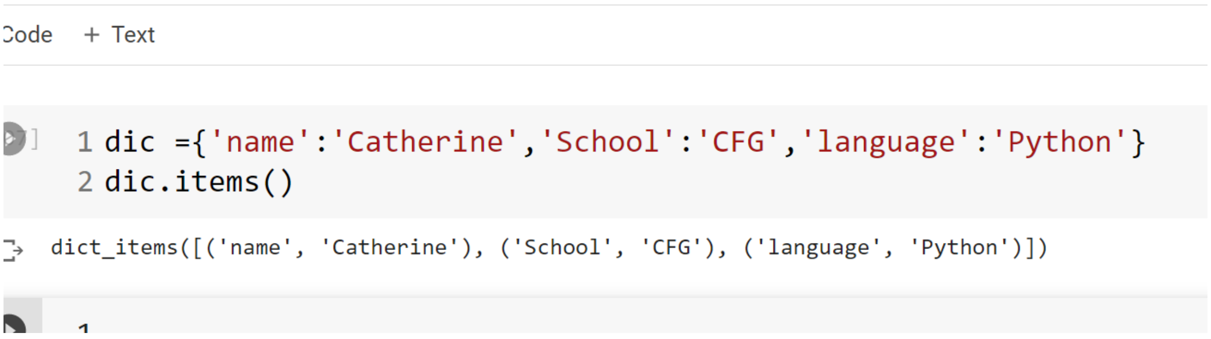
get():returns the value of the item with the specified key



```python
dic ={'name':'Catherine','School':'CFG','language':'Python'}
dic.get('name')
```

```
'Catherine'
```

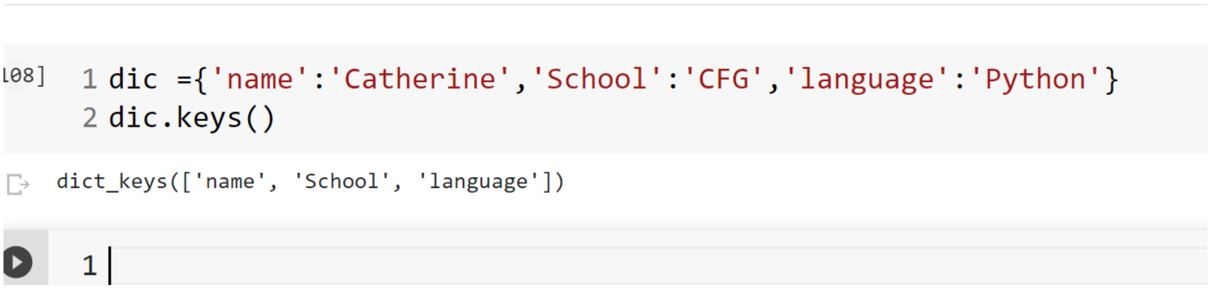
items(): returns a view object. The view object contains the key-value pairs of the dictionary



```python
dic ={'name':'Catherine','School':'CFG','language':'Python'}
dic.items()
```

```
dict_items([('name', 'Catherine'), ('School', 'CFG'), ('language', 'Python')])
```

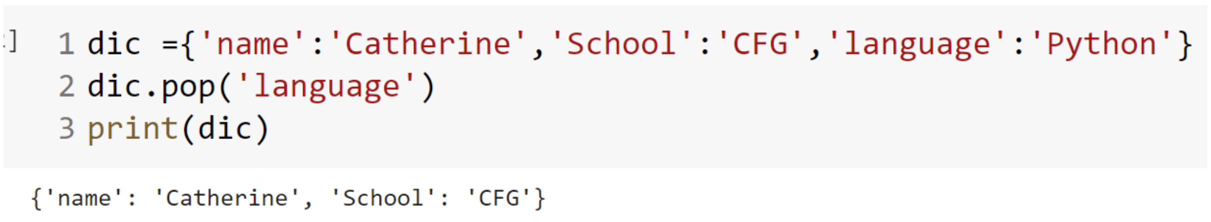
keys(): get the keys from a dict



```python
dic ={'name':'Catherine','School':'CFG','language':'Python'}
dic.keys()
```

```
dict_keys(['name', 'School', 'language'])
```

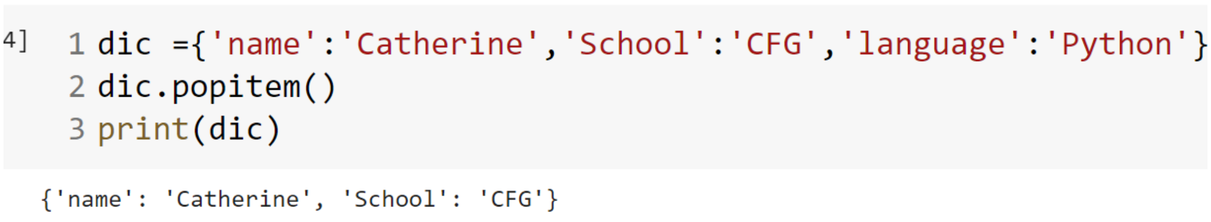
pop(): delete the element at the specified position



```python
dic ={'name':'Catherine','School':'CFG','language':'Python'}
dic.pop('language')
print(dic)
```

```
{'name': 'Catherine', 'School': 'CFG'}
```

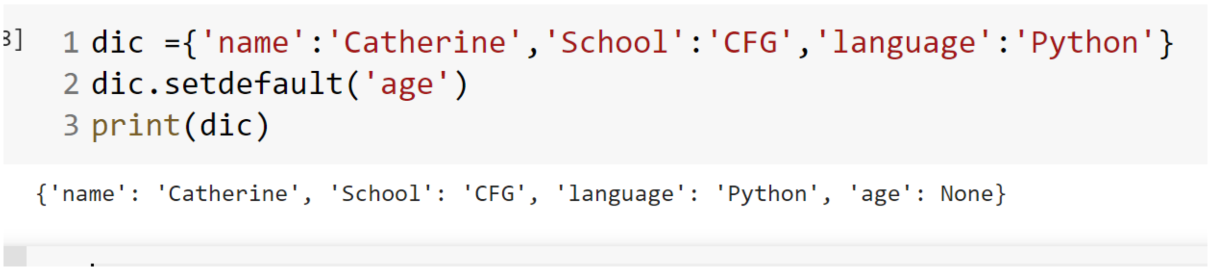
popitem():removes the item that was last inserted into the dictionary



```python
dic ={'name':'Catherine','School':'CFG','language':'Python'}
dic.popitem()
print(dic)
```

```
{'name': 'Catherine', 'School': 'CFG'}
```

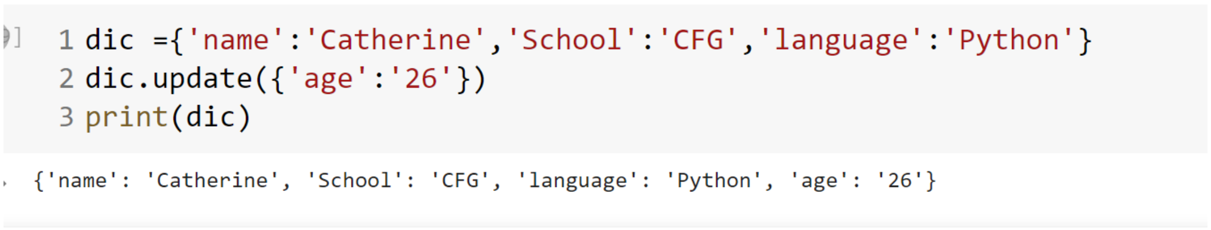
setdefault():returns the value of the item with the specified key.If the key does not exist, insert the key, with the specified value



```python
dic ={'name':'Catherine','School':'CFG','language':'Python'}
dic.setdefault('age')
print(dic)
```

```
{'name': 'Catherine', 'School': 'CFG', 'language': 'Python', 'age': None}
```

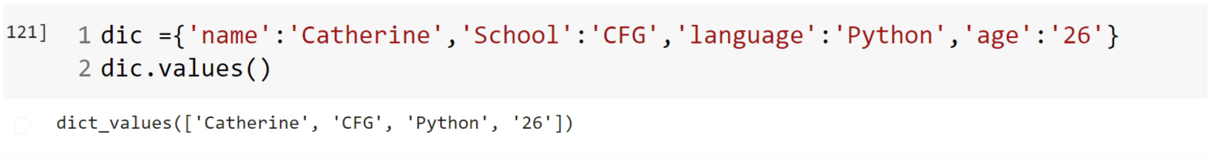
update(): insert the items to dict



```python
dic ={'name':'Catherine','School':'CFG','language':'Python'}
dic.update({'age':'26'})
print(dic)
```

```
{'name': 'Catherine', 'School': 'CFG', 'language': 'Python', 'age': '26'}
```

values(): reflect any changes done to the dict

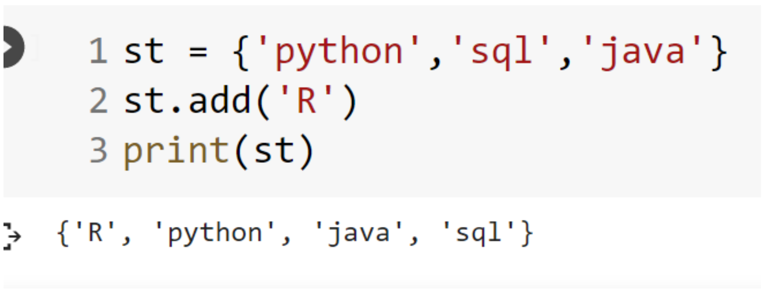


```python
dic ={'name':'Catherine','School':'CFG','language':'Python','age':'26'}
dic.values()
```

```
dict_values(['Catherine', 'CFG', 'Python', '26'])
```

Python set methods:

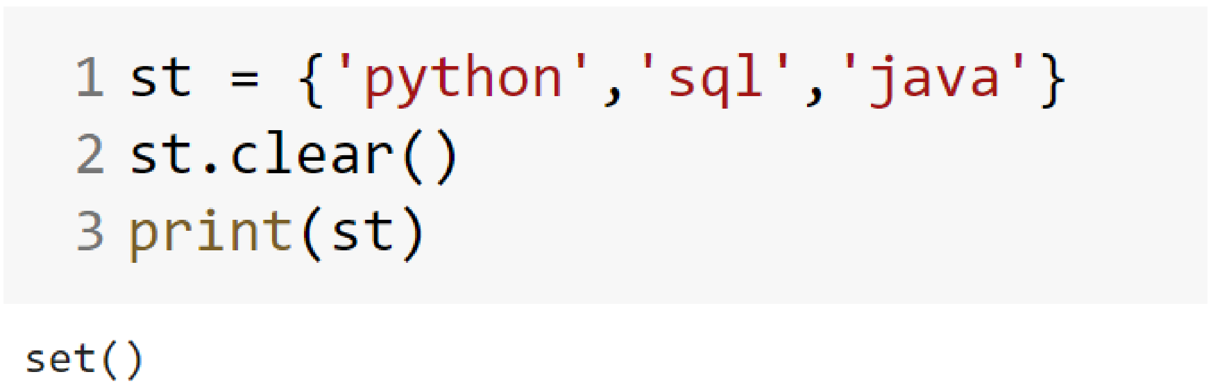
add(): adds an element to the set. If the element already exists, it does not add the element



```
st = {'python','sql','java'}
st.add('R')
print(st)
```

```
{'R', 'python', 'java', 'sql'}
```

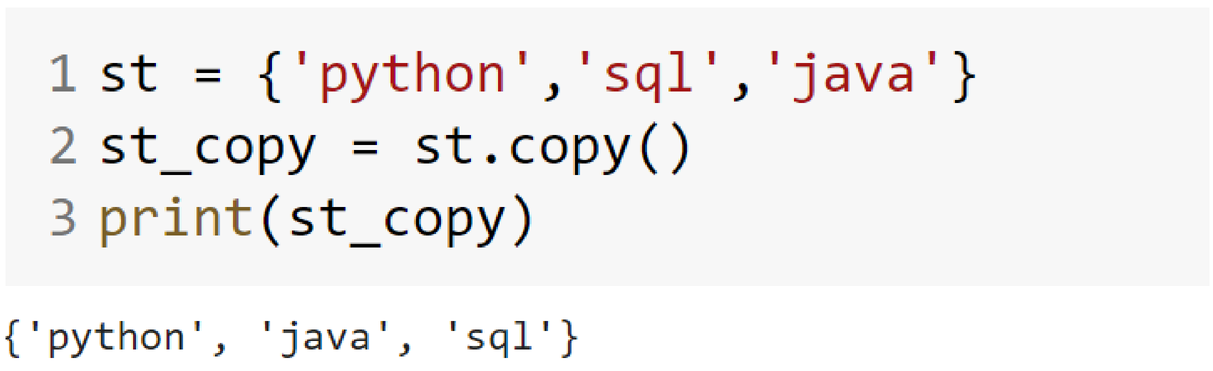
clear(): clear everything in a set



```
st = {'python','sql','java'}
st.clear()
print(st)
```

```
set()
```

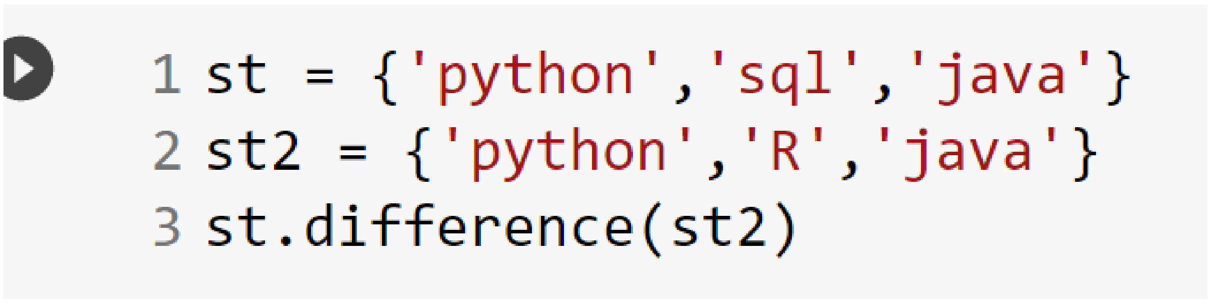
copy(): make a copy of the set



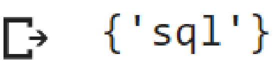
```
st = {'python','sql','java'}
st_copy = st.copy()
print(st_copy)
```

```
{'python', 'java', 'sql'}
```

difference():returns a set that contains the difference between two sets

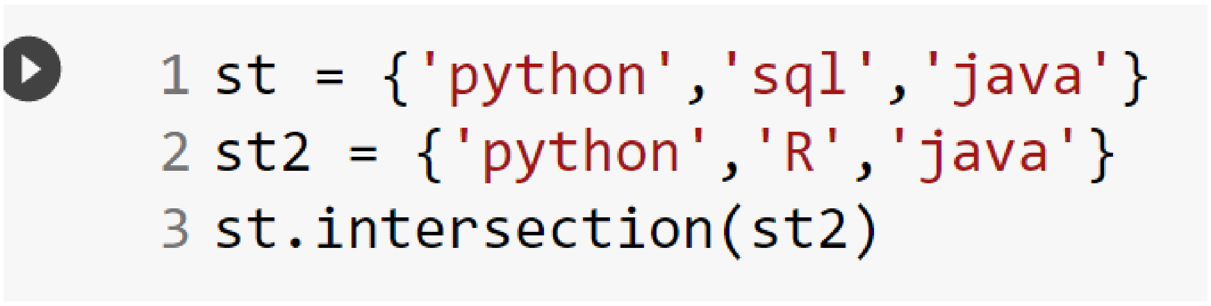


```python
st = {'python','sql','java'}
st2 = {'python','R','java'}
st.difference(st2)
```

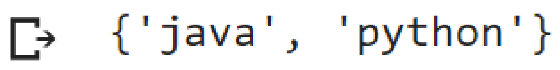


```
{'sql'}
```

intersection(): returns a set that contains the common part between two sets

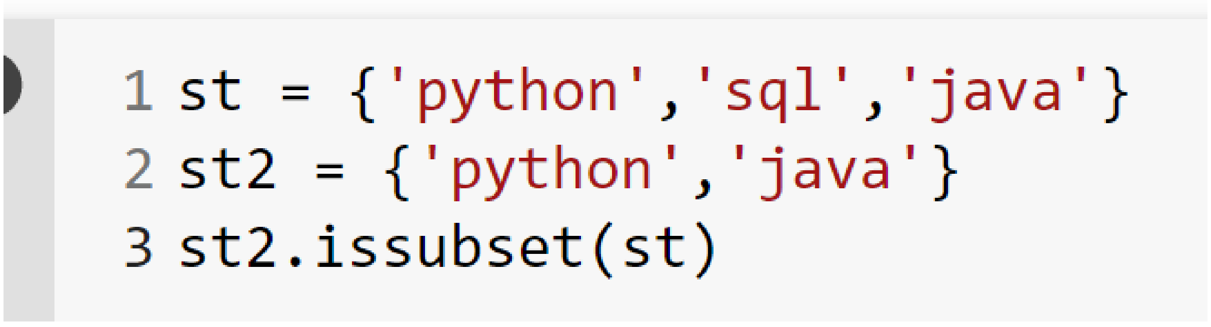


```python
st = {'python','sql','java'}
st2 = {'python','R','java'}
st.intersection(st2)
```

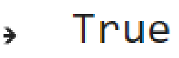


```
{'java', 'python'}
```

issubset(): returns True if all items in the set exists in the specified set, otherwise return False

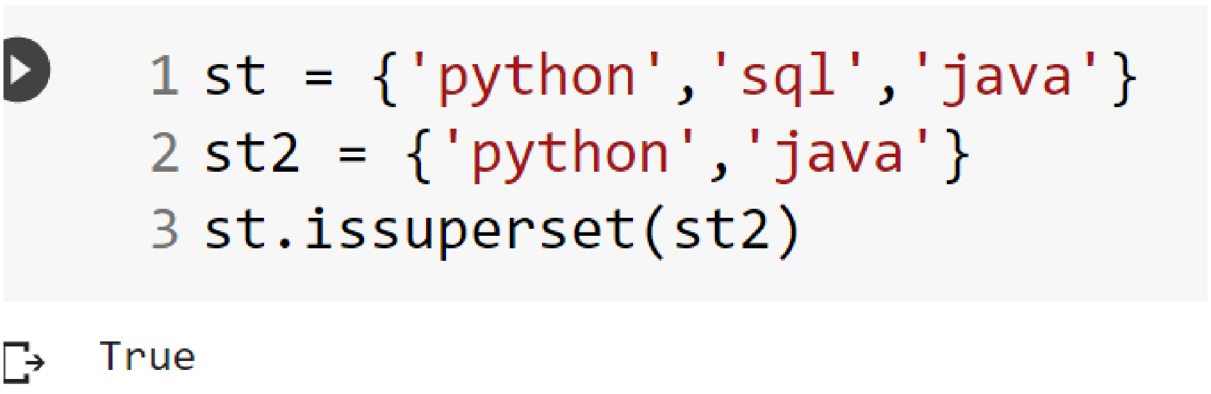


```python
st = {'python','sql','java'}
st2 = {'python','java'}
st2.issubset(st)
```



```
True
```

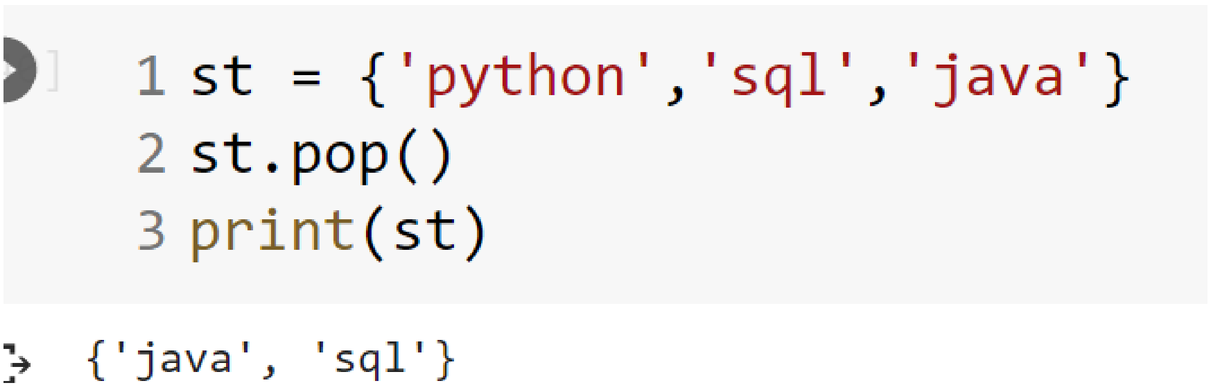
issuperset():returns True if all items in the specified set exists in the original set, otherwise return False



```python
st = {'python','sql','java'}
st2 = {'python','java'}
st.issuperset(st2)
```

```
True
```

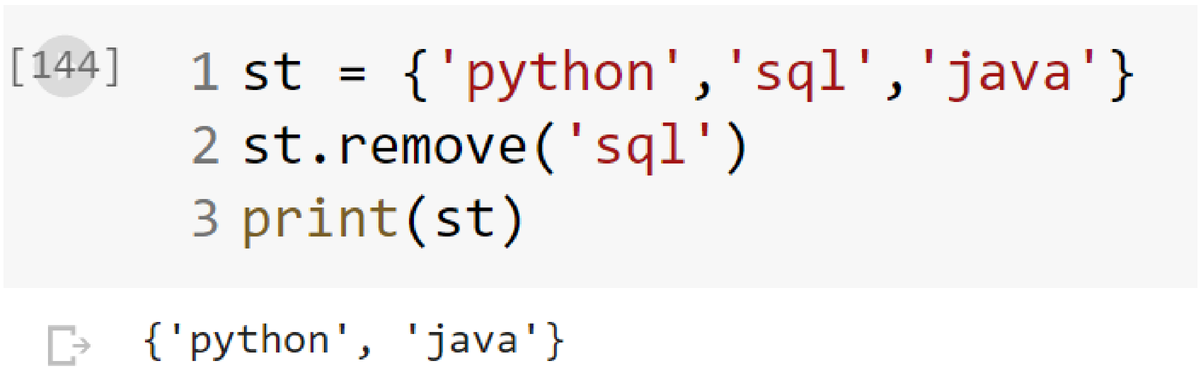
pop():returns the removed item



```python
st = {'python','sql','java'}
st.pop()
print(st)
```

```
{'java', 'sql'}
```

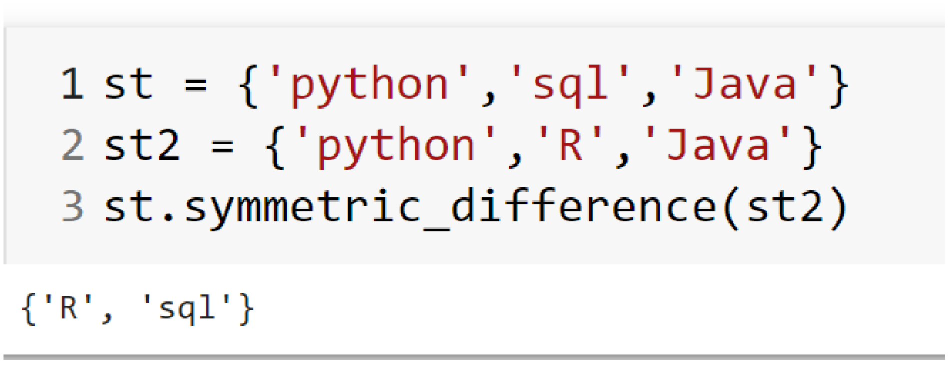
remove():removes the first occurrence of the element with the specified value



```python
st = {'python','sql','java'}
st.remove('sql')
print(st)
```

```
{'python', 'java'}
```

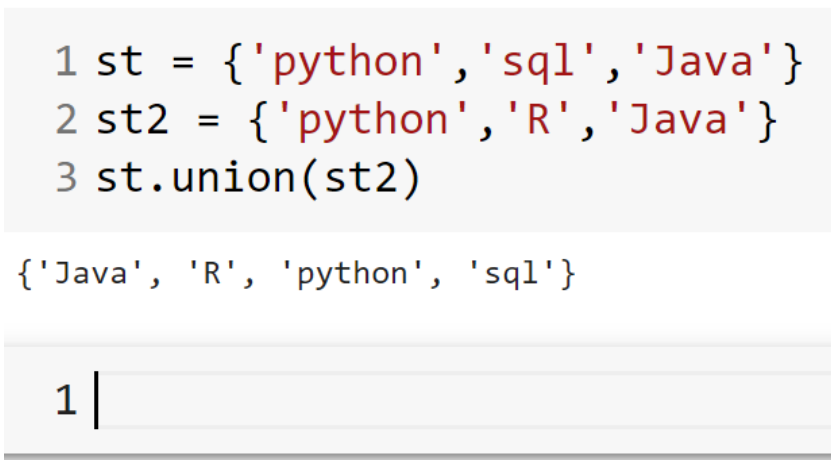
symmetric_difference(): returns a set that contains all items from both set, but not the items that are present in both sets.



```python
st = {'python','sql','Java'}
st2 = {'python','R','Java'}
st.symmetric_difference(st2)
```

```
{'R', 'sql'}
```

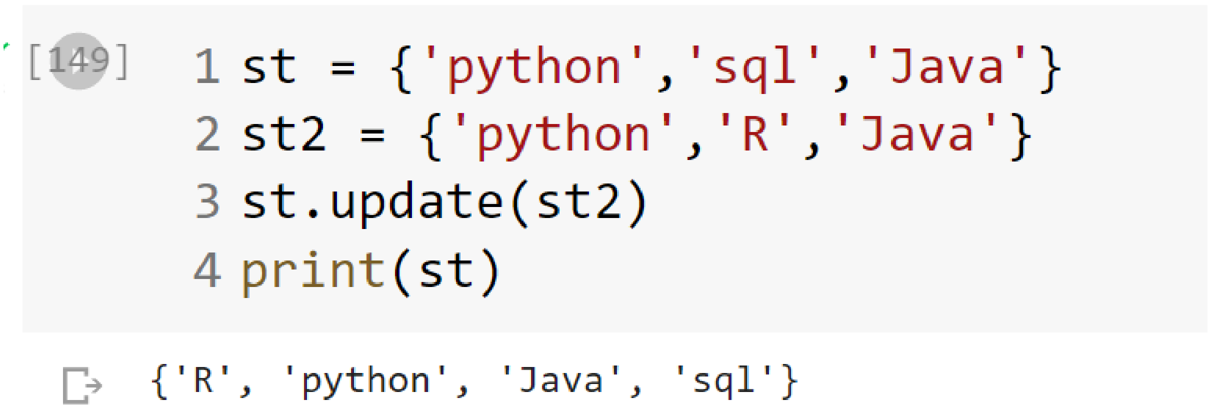
union():a set that contains all items from the original set, and all items from the specified set(s)



```python
st = {'python','sql','Java'}
st2 = {'python','R','Java'}
st.union(st2)
```

```
{'Java', 'R', 'python', 'sql'}
```

update():updates the current set, by adding items from another set (or any other iterable

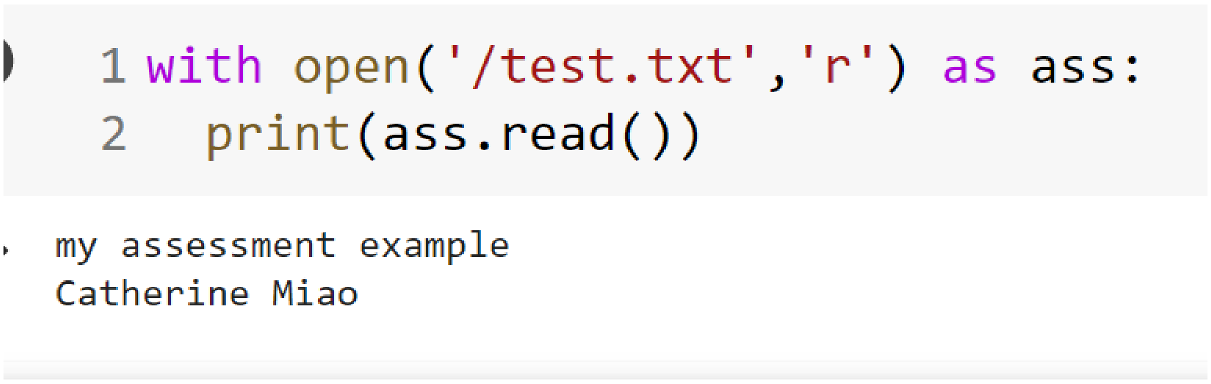


```python
st = {'python','sql','Java'}
st2 = {'python','R','Java'}
st.update(st2)
print(st)
```

```
{'R', 'python', 'Java', 'sql'}
```

Python file methods:

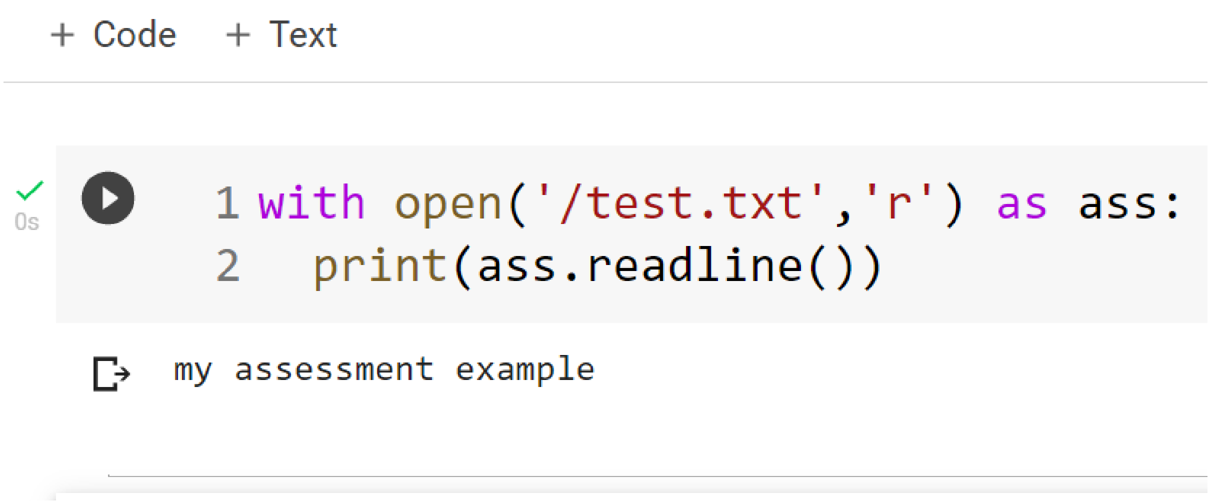
read():reading the content of a file



```python
with open('/test.txt','r') as ass:
  print(ass.read())
```

```
my assessment example
Catherine Miao
```

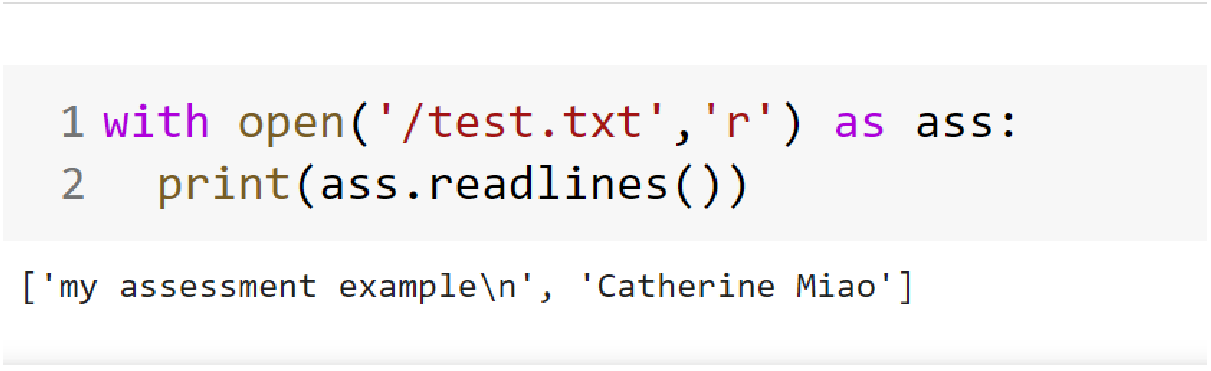
readline(): return one line from the file



```python
with open('/test.txt','r') as ass:
  print(ass.readline())
```

```
my assessment example
```

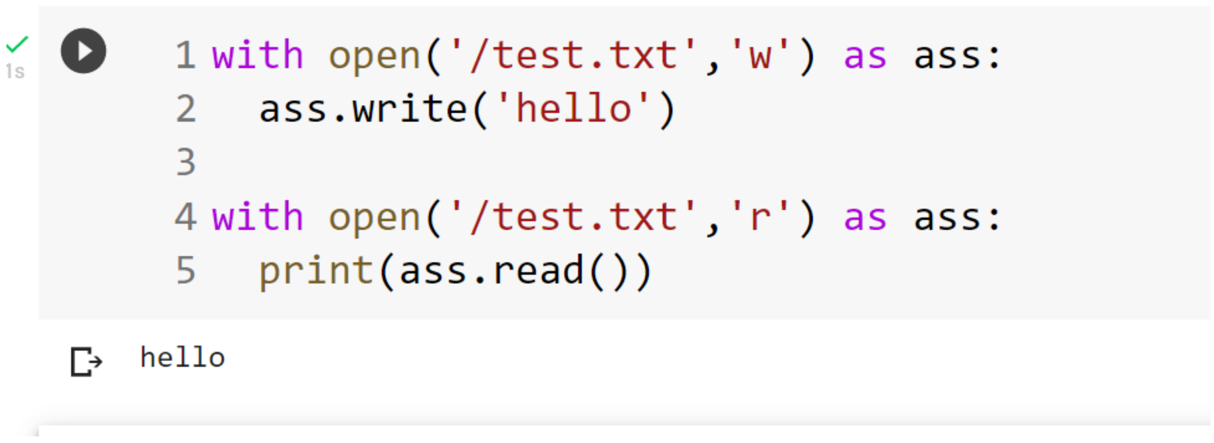
readlines():returns a list containing each line in the file as a list item



```python
with open('/test.txt','r') as ass:
  print(ass.readlines())
```

```
['my assessment example\n', 'Catherine Miao']
```

write():to write a file, write() overwrite any existing content

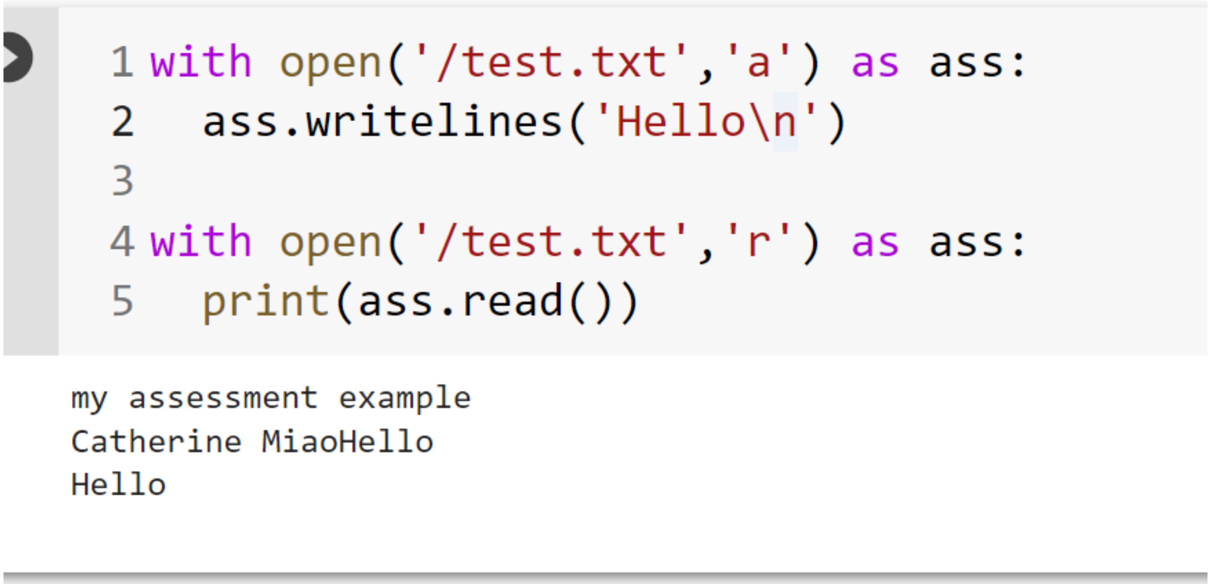


```python
with open('/test.txt','w') as ass:
  ass.write('hello')

with open('/test.txt','r') as ass:
  print(ass.read())
```

```
hello
```

writelines():writes the items of a list to the file.#



```python
with open('/test.txt','a') as ass:
  ass.writelines('Hello\n')

with open('/test.txt','r') as ass:
  print(ass.read())
```

```
my assessment example
Catherine MiaoHello
Hello
```