EXPERIMENT NO : 2C

Python Programs To Implement Built-in Set and String Functions

NAME : AKASH RAMKRIT YADAV ID.NO: VU4F2122016

BATCH : A BRANCH : IT DIV : A

Aim :- Python Programs To Implement Built-in Set and . . String functions

THEORY:

OUTPUT:

```
Python 3.11.0a4 (main, Jan 17 2022, 12:57:32) [MSC v.1929 32 bit
(Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more
information.
#AKASH RAMKRIT YADAV   #ID NO:VU4F2122016    DATE:31/01/2023
```

SET OPERATIONS:

1#Python Set add() Method

#Definition and Usage

The add() method adds an element to the set.

If the element already exists, the add() method does not add the element.

Syntax
set.add(elmnt)

Parameter Values
Parameter Description
Element Required. The element to add to the set

#Create a Set:

```
>>name={"akash","suraj","viram","suny","suray","saurabh"}
  name.add("kush")
  print(name)
  {'suny', 'kush', 'suray', 'saurabh', 'suraj', 'akash',     .
. 'viram'}
```

2#Python Set clear() Method

#Definition and Usage
The clear() method removes all elements in a set.

Syntax
set.clear()

#Create a Set:

```
>>name={"akash","suraj","viram","suny","suray","saurabh"}
  print(name)
  {'suny', 'suray', 'saurabh', 'suraj', 'akash', 'viram'}
  name.clear()
  print(name)
  set()
```

3#Python Set copy() Method

#Definition and Usage
The copy() method copies the set.

Syntax
set.copy()

#Create a Set:

```
>>name={"akash","suraj","viram","suny","suray","saurabh"}

  a1= name.copy()
  print(a1)
  {'suny', 'suray', 'saurabh', 'suraj', 'akash', 'viram'}
```

==========

```
name={"akash","suraj","viram","suny","suray","saurabh"}

a1= name
print(a1)
{'suny', 'suray', 'saurabh', 'suraj', 'akash', 'viram'}
```

==========

4#Python Set difference() Method

#Definition and Usage

The difference() method returns a set that contains the difference between two sets.

Meaning: The returned set contains items that exist only in the first set, and not in both sets.

Syntax
set.difference(set)

#Create a Set:

```
>>name={"akash","suraj","viram","suny","suray","saurabh"}
  company={"google", "microsoft", "akash"}
  a1a=name.difference(company)
  print(a1a)
  {'suray', 'saurabh', 'viram', 'suny', 'suraj'}
```

5#Python Set difference_update() Method

#Definition and Usage

The difference_update() method removes the items that exist in both sets.

The difference_update() method is different from the difference() method, because the difference() method returns a new set, without the unwanted items, and the difference_update() method removes the unwanted items from the original set.

Syntax
set.difference_update(set)

#Create a Set:

```
>>name={"akash","suraj","viram","suny","suray","saurabh"}

  company={"google", "microsoft", "akash"}

  name.difference_update(company)
  print(name)
  {'suny', 'suray', 'saurabh', 'suraj', 'viram'}
```

6#Python Set discard() Method

#Definition and Usage

The discard() method removes the specified item from the set.

This method is different from the remove() method, because the remove() method will raise an error if the specified item does not exist, and the discard() method will not.

Syntax
set.discard(value)

#Create a Set:

```
>>name={"akash","suraj","viram","suny","suray","saurabh"}
  name.discard("saurabh")
  print(name)
  {'suny', 'suray', 'suraj', 'akash', 'viram'}
```

7#Python Set intersection() Method

#Definition and Usage

The intersection() method returns a set that contains the similarity between two or more sets.

Meaning: The returned set contains only items that exist in both sets, or in all sets if the comparison is done with more than two sets.

Syntax

```
set.intersection(set1, set2 ... etc)
```

#Create a Set:

```
>>name={"akash","suraj","viram","suny","suray","saurabh"}
  company={"google", "microsoft", "akash"}
  a1=name.intersection(company)
  print(a1)
  {'akash'}
```

8#Python Set intersection_update() Method

#Definition and Usage

The intersection_update() method removes the items that is not present in both sets (or in all sets if the comparison is done between more than two sets).

The intersection_update() method is different from the intersection() method, because the intersection() method returns a new set, without the unwanted items, and the intersection_update() method removes the unwanted items from the original set.

Syntax

```
set.intersection_update(set1, set2 ... etc)
```

#Create a Set:

```
>>name={"akash","suraj","viram","suny","suray","saurabh"}

  company={"google", "microsoft", "akash"}
  name.intersection_update(company)
  print(name)
  {'akash'}
```

9#Python Set isdisjoint() Method

#Definition and Usage

The isdisjoint() method returns True if none of the items are present in both sets, otherwise it returns False.

Syntax
set.isdisjoint(set)

#Create a Set:

```
>>name={"akash","suraj","viram","suny","suray","saurabh"}
  company={"google", "microsoft", "ibm"}
  a1=name.isdisjoint(company)
  print(a1)
  True
```

10#Python Set issubset() Method

#Definition and Usage

The issubset() method returns True if all items in the set exists in the specified set, otherwise it returns False.

Syntax
set.issubset(set)\

#Create a Set:

```
a1={'1',"2","3"}
a2={"6","7","1","2","3","4","8"}
a=a1.issubset(a2)
print(a)
True
```

11#Python Set issuperset() Method

#Definition and Usage

The issuperset() method returns True if all items in the specified set exists in the original set, otherwise it returns False.

Syntax
set.issuperset(set)

#Create a Set:

```
>>a1={"6","7","1","2","3","4","8"}
  a2={'1',"2","3"}
  a=a1.issuperset(a2)
  print(a)
```

```
True
```

12#Python Set pop() Method

#Definition and Usage

The pop() method removes a random item from the set.

This method returns the removed item.

Syntax
set.pop()

#Create a Set:

```
>>name={"akash","suraj","viram","suny","suray","saurabh"}
  name.pop()
  'suny'
   print(name)
  {'suray', 'saurabh', 'suraj', 'akash', 'viram'}
```

13#Python Set remove() Method

#Definition and Usage

The remove() method removes the specified element from the set.

This method is different from the discard() method, because the remove() method will raise an error if the specified item does not exist, and the discard() method will not.

Syntax
set.remove(item)

#Create a Set:

```
>>name={"akash","suraj","viram","suny","suray","saurabh"}
  name.remove("akash")
  print(name)
  {'suny', 'suray', 'saurabh', 'suraj', 'viram'}
```

14#Python Set symmetric_difference() Method

#Definition and Usage

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

Meaning: The returned set contains a mix of items that are not present in both sets.

Syntax
set.symmetric_difference(set)

#Create a Set:

```
>>name={"akash","suraj","viram","suny","suray","saurabh"}
  company={"google", "microsoft", "ibm"}
  a=name.symmetric_difference(company)
  print(a)
  {'suray', 'ibm', 'saurabh', 'suraj', 'google', 'viram', 'suny',
'microsoft', 'akash'}
```

15#Python Set union() Method

#Definition and Usage

The union() method returns a set that contains all items from the original set, and all items from the specified set(s).

You can specify as many sets you want, separated by commas.

It does not have to be a set, it can be any iterable object.

If an item is present in more than one set, the result will contain only one appearance of this item.

Syntax
set.union(set1, set2...)

#Create a Set:

```
>>name={"akash","suraj","viram","suny","suray","saurabh"}
  company={"google", "microsoft", "akash"}
  a=name.union(company)
  print(a)
  {'suray', 'saurabh', 'microsoft', 'google', 'viram', 'suny',
'suraj', 'akash'}
```

16#Python Set update() Method

#Definition and Usage

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

If an item is present in both sets, only one appearance of this item will be present in the updated set.

Syntax
set.update(set)

#Create a Set:

```
>>name={"akash","suraj","viram","suny","suray","saurabh"}
  company={"google", "microsoft", "ibm"}
```

```
  name.update(company)
  print(name)
  {'suray', 'ibm', 'saurabh', 'microsoft', 'google', 'viram',
'suny', 'suraj', 'akash'}
```

#STRING FUNCTIONS

1#Python String encode() Method

#Definition and Usage

The encode() method encodes the string, using the specified encoding. If no encoding is specified, UTF-8 will be used.

Syntax
string.encode(encoding=encoding, errors=errors)

#Create a Set:

```
>>a1="MY NAME IS AKASH YADAV"
  x=a1.encode()
  print(x)
  b'MY NAME IS AKASH YADAV'
```

2#Python String endswith() Method

#Definition and Usage

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

Syntax
string.endswith(value, start, end)

#Create a Set:

```
>>a1="MY NAME IS AKASH YADAV"
  x=a1.endswith("AKASH YADAV")
  print(x)
  True
  x1=a1.endswith("YADAV")
  print(x1)
  True
  x2=a1.endswith("yadav")
  print(x2)
  False
```

3#Python String expandtabs() Method

#Definition and Usage

The expandtabs() method sets the tab size to the specified number of whitespaces.

Syntax
string.expandtabs(tabsize)
Parameter Values
Parameter Description
tabsize Optional. A number specifying the tabsize. Default tabsize is 8

#Create a Variable:

```
>>a1="HI\tAKASH\tRAMKRIT\tYADAV"
  print(a1.expandtabs())
  HI      AKASH   RAMKRIT YADAV
  print(a1.expandtabs(2))
  HI  AKASH RAMKRIT YADAV
  print(a1.expandtabs(6))
  HI    AKASH RAMKRIT     YADAV
```

4#Python String find() Method

#Definition and Usage

The find() method finds the first occurrence of the specified value.

The find() method returns -1 if the value is not found.

The find() method is almost the same as the index() method, the only difference is that the index() method raises an exception if the value is not found. (See example below)

Syntax
string.find(value, start, end)
Parameter Values
Parameter Description
value Required. The value to search for
start Optional. Where to start the search. Default is 0
... end Optional. Where to end the search. Default is to the end of the string

#Create a Variable:

```
>> a1="hello,welcom to python world"
   x=a1.find("welcome")
   print(x)
   -1
>>> x1=a1.find("to")
>>> print(x1)
13
>>> x2=a1.find("world")
>>> print(x2)
```

5#Python String format() Method

#Definition and Usage

The format() method formats the specified value(s) and insert them inside the string's placeholder.

The placeholder is defined using curly brackets: {}. Read more about the placeholders in the Placeholder section below.

The format() method returns the formatted string.

Syntax
string.format(value1, value2...)
Parameter Values
Parameter Description
value1, value2... Required. One or more values that should be formatted and inserted in the string.

The values are either a list of values separated by commas, a key=value list, or a combination of both.

#Create a Variable:

```
>>a1="one day income of akash yadav {price:.2f} dollars!"
  print(a1.format(price=100000000))
  one day income of akash yadav 100000000.00 dollars!
```

6#Python String format_map() Method

#Python String format_map() method is an inbuilt function in Python, which is used to return a dictionary key's value.

Syntax:

string.format_map(z)

#Create a Set:

```
>>a={'A':'AKASH','B':'YADAV'}
  print("{A}'s SURNAME IS {B}".format_map(a))
  AKASH's SURNAME IS YADAV
  a={'A':'AKASH','B':'SUARAJ,VIRAM,SURAYA,SUNNY'}
  print("{A}'S friends are {B}".format_map(a))
  AKASH'S friends are SUARAJ,VIRAM,SURAYA,SUNNY
```

7#Python String index() Method

#Definition and Usage

The index() method finds the first occurrence of the specified value.

The index() method raises an exception if the value is not found.

The index() method is almost the same as the find() method, the only difference is that the find() method returns -1 if the value is not found. (See example below)

Syntax
string.index(value, start, end)

#Create a Variable:

```
>>a1="HELLO!, SIR HOW ARE YOU!"
  print(a1.index("S"))
  8
  print(a1.index("SIR"))
  8
  print(a1.index("Y"))
  20
```

8#Python String isalnum() Method

#Definition and Usage

The isalnum() method returns True if all the characters are alphanumeric, meaning alphabet letter (a-z) and numbers (0-9).

Example of characters that are not alphanumeric: (space)!#%&? etc.

Syntax
string.isalnum()

#Create a Variable:

```
a1="AKASH ID NO IS VU4F2122016"
print(a1.isalnum())
False

a2=("AKASH12")
print(a2.isalnum())
True

a3="akash yadav"
print(a)
{'A': 'AKASH', 'B': 'SUARAJ,VIRAM,SURAYA,SUNNY'}
a3="akash yadav"
print(a3.isalnum())
False
```

9#Python String isalpha() Method

#Definition and Usage

The isalpha() method returns True if all the characters are alphabet letters (a-z).

Example of characters that are not alphabet letters: (space)!#%&? etc.

Syntax
string.isalpha()

#Create a Variable:

```
a1="my name is akash yadav"

print(a1.isalpha())
```

False

```
b1="MY NAME AKASH YADAV"
print(b1.isalpha())
```

False

```
a1="AKASH"
print(a1.isalpha())
```

True

```
a1="AKASH12"
print(a1.isalpha())
```

False

10#Python String isascii() Method

#Definition and Usage

The isascii() method returns True if all the characters are ascii characters (a-z).

Check our ASCII Reference.

Syntax
string.isascii()

#Create a Variable:

```
a1="AKASH123"
print(a1.isascii())
```

True

11#Python String isdecimal() Method

#Definition and Usage

The isdecimal() method returns True if all the characters are decimals (0-9).

This method is used on unicode objects.

Syntax
string.isdecimal()

#Create a Variable:

```
a1="101"
print(a1.isdecimal())
True
```

12#Python String isdigit() Method

#Definition and Usage

The isdigit() method returns True if all the characters are digits, otherwise False.

Exponents, like ², are also considered to be a digit.

Syntax
string.isdigit()

#Create a Variable:

```
a1="43268468"

print(a1.isdigit())
True

a1="akash"
print(a1.isdigit())
False
```

13#Python String isidentifier() Method

#Definition and Usage

The isidentifier() method returns True if the string is a valid identifier, otherwise False.

A string is considered a valid identifier if it only contains alphanumeric letters (a-z) and (0-9), or underscores (_). A valid identifier cannot start with a number, or contain any spaces.

Syntax
string.isidentifier()

#Create a Variable:

```
a1="AKASH_123"
print(a1.isidentifier())
True

a1="akash@123"
print(a1.isidentifier())
False
```

14#Python String islower() Method

#Definition and Usage

The islower() method returns True if all the characters are in lower case, otherwise False.

Numbers, symbols and spaces are not checked, only alphabet characters.

Syntax
string.islower()

#Create a Variable:

```
a1="my name is akash yadav did you know!"
print(a1.islower())
True

a1="my name is AKASH YADAV"
print(a1.islower())
False
```

15#Python String isnumeric() Method

#Definition and Usage

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

Exponents, like ² and ¾ are also considered to be numeric values.

"-1" and "1.5" are NOT considered numeric values, because all the characters in the string must be numeric, and the - and the . are not.

#Syntax
string.isnumeric()

#Create a Variable:

```
a1="101"
print(a1.isnumeric())
True

a1="1.23"
print(a1.isnumeric())
False
```

16#Python String isprintable() Method

#Definition and Usage

The isprintable() method returns True if all the characters are printable, otherwise False.

Example of none printable character can be carriage return and line feed.

Syntax
string.isprintable()

#Create a Variable:

```
a1="akash yadav is my name!!"
x=a1.isprintable()
print(x)
True
```

17#Python String isspace() Method

#Definition and Usage

The isspace() method returns True if all the characters in a string are whitespaces, otherwise False.

Syntax
string.isspace()

#Create a Variable:

```
a1="  "

print(a1.isspace())
True

a1 ="  a "
print(a1.isspace())
False
```

18#Python String istitle() Method

#Definition and Usage

The istitle() method 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.

Symbols and numbers are ignored.

Syntax
string.istitle()

#Create a Variable:

```
a1="Wlcome! To My World!"
>>> print(a1.istitle())
True
```

19#Python String isupper() Method

#Definition and Usage

The isupper() method returns True if all the characters are in upper case, otherwise False.

Numbers, symbols and spaces are not checked, only alphabet characters.

Syntax
string.isupper()

#Create a Variable:

```
a1="AKASH YADAV"
print(a1.isupper())
True

a1="AKASH yadav"
print(a1.isupper)
<built-in method isupper of str object at 0x0000026D733F5AB0>
```

20#Python String join() Method

#Definition and Usage

The join() method takes all items in an iterable and joins them into one string.

A string must be specified as the separator.

Syntax
string.join(iterable)

#Create a Variable:

```
a1=("akash","suraj","viram")
X="YADAV".join(a1)
print(X)
akashYADAVsurajYADAVviram


a2="_YADAV\n".join(a1)
print(a2)
```

```
akash_YADAV
suraj_YADAV
viram
```

21#Python String ljust() Method

#Definition and Usage

The ljust() method will left align the string, using a specified character (space is default) as the fill character.

Syntax
string.ljust(length, character

#Create a Variable:

```
a1="AKASH"

X=a1.ljust(10)

print(X,"is my name")
```

```
AKASH      is my name
```

22#Python String lower() Method

#Definition and Usage

The lower() method returns a string where all characters are lower case.

Symbols and Numbers are ignored.

Syntax
string.lower()

#Create a Variable:

```
a1="AKASH Yadav "

print(a1.lower())
```

```
akash yadav
```

23#Python String lstrip() Method

#Definition and Usage

The lstrip() method removes any leading characters (space is the default leading character to remove)

Syntax
string.lstrip(characters)

#Create a Variable:

```
a1="  hi   "

x=a1.lstrip()

print("all of you",x," how are you")
```

```
all of you hi     how are you
```

24#Python String partition() Method

#Definition and Usage

The partition() method searches for a specified string, and splits the string into a tuple containing three elements.

The first element contains the part before the specified string.

The second element contains the specified string.

The third element contains the part after the string.

Note: This method searches for the first occurrence of the specified string.

Syntax
```
string.partition(value)
```

#Create a Variable:

```
a1="every BOY should have GF"

x=a1.partition("BOY")

print(x)
```

```
('every ', 'BOY', ' should have GF')
```

25#Python String replace() Method

#Definition and Usage

The replace() method replaces a specified phrase with another specified phrase.

Note: All occurrences of the specified phrase will be replaced, if nothing else is specified.

Syntax
```
string.replace(oldvalue, newvalue, count)
```

#Create a Variable:

```
>>a1="I LIKE MANGOES"

  X=a1.replace("MANGOES","apple")

  print()
  KeyboardInterrupt
  print(X)

  I LIKE apple
```

26#Python String rfind() Method

#Definition and Usage

The rfind() method finds the last occurrence of the specified value.

The rfind() method returns -1 if the value is not found.
The rfind() method is almost the same as the rindex() method. See example below.

Syntax
string.rfind(value, start, end)

```
a1=("my name is akash")
x=a1.rfind("akash")
print(x)

11
```

27#Python String swapcase() Method

#Definition and Usage

The swapcase() method returns a string where all the upper case letters are lower case and vice versa.

Syntax
string.swapcase()

#Create a Variable:

```
>>a1="MY NAME IS akash yadav"

   x=a1.swapcase()

   print(x)

   my name is AKASH YADAV
```

28#Python String upper() Method

#Definition and Usage

The upper() method returns a string where all characters are in upper case.

Symbols and Numbers are ignored.

Syntax
string.upper()

#Create a Variable:

```
>>a1="MY name is akash"

  print(a1.upper())

  MY NAME IS AKASH
```

29#Python String title() Method

#Definition and Usage

The title() method returns a string where the first character in every word is upper case. Like a header, or a title.

If the word contains a number or a symbol, the first letter after that will be converted to upper case.

Syntax
string.title()

#Create a Variable:

```
>>a1=("welcome to pvppcoe college of enginnering")

  print(a1.title())

  Welcome To Pvppcoe College Of Enginnering
```