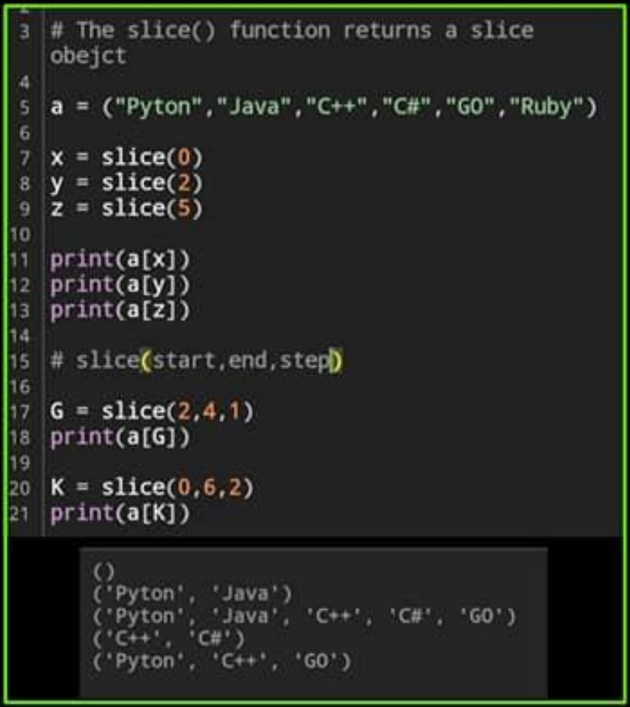
Python

Built-in Functions

Amir Sakib Saad

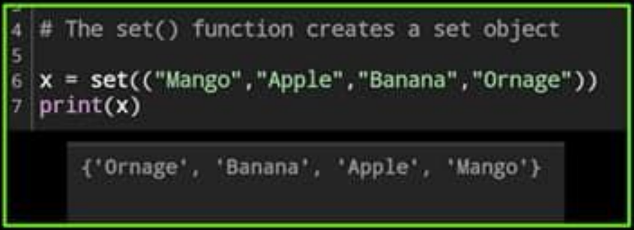
slice( )



```python
# The slice() function returns a slice
obejct

a = ("Pyton","Java","C++","C#","GO","Ruby")

x = slice(0)
y = slice(2)
z = slice(5)

print(a[x])
print(a[y])
print(a[z])

# slice(start,end,step)

G = slice(2,4,1)
print(a[G])

K = slice(0,6,2)
print(a[K])
```

```
()
('Pyton', 'Java')
('Pyton', 'Java', 'C++', 'C#', 'GO')
('C++', 'C#')
('Pyton', 'C++', 'GO')
```

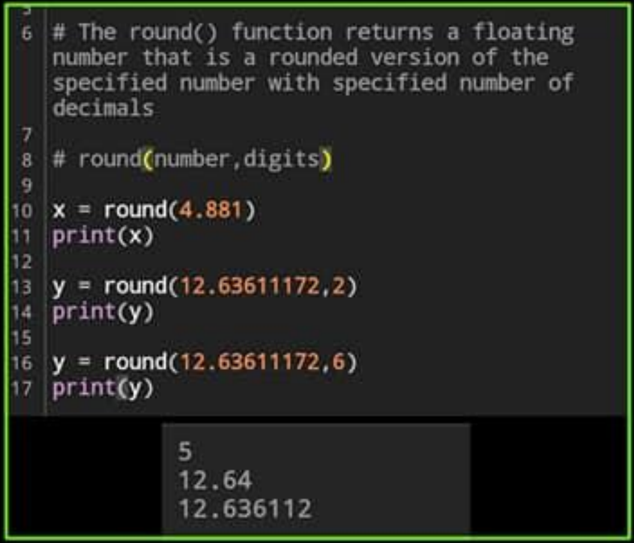
set( )



```python
# The set() function creates a set object

x = set(("Mango","Apple","Banana","Ornage"))
print(x)
```

```
{'Ornage', 'Banana', 'Apple', 'Mango'}
```

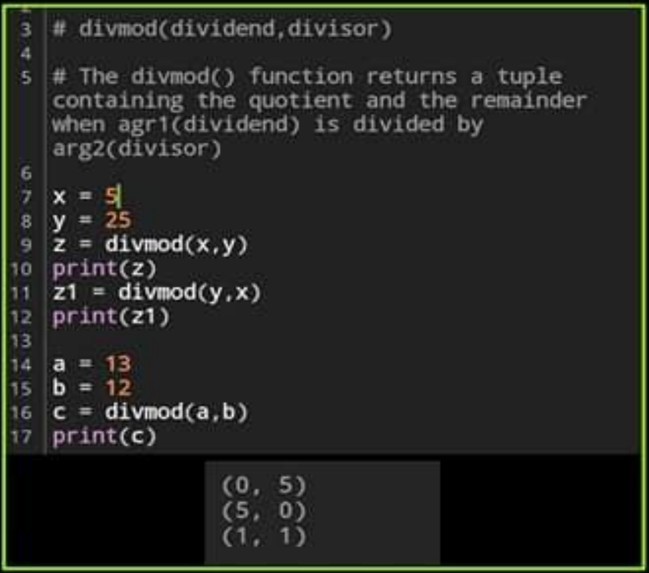
round( )



```python
# The round() function returns a floating
number that is a rounded version of the
specified number with specified number of
decimals

# round(number,digits)

x = round(4.881)
print(x)

y = round(12.63611172,2)
print(y)

y = round(12.63611172,6)
print(y)
```

```
5
12.64
12.636112
```

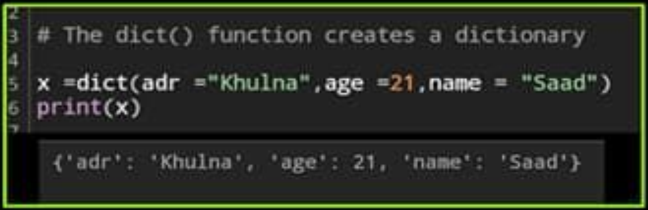
divmod( )



```python
# divmod(dividend,divisor)

# The divmod() function returns a tuple containing the quotient and the remainder when agr1(dividend) is divided by arg2(divisor)

x = 5
y = 25
z = divmod(x,y)
print(z)
z1 = divmod(y,x)
print(z1)

a = 13
b = 12
c = divmod(a,b)
print(c)
```

```
(0, 5)
(5, 0)
(1, 1)
```

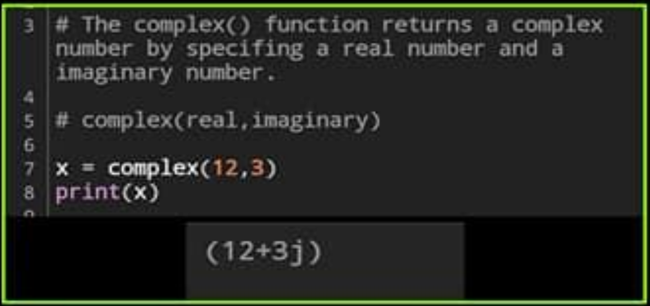
dict( )



```python
# The dict() function creates a dictionary

x =dict(adr ="Khulna",age =21,name = "Saad")
print(x)
```

```
{'adr': 'Khulna', 'age': 21, 'name': 'Saad'}
```

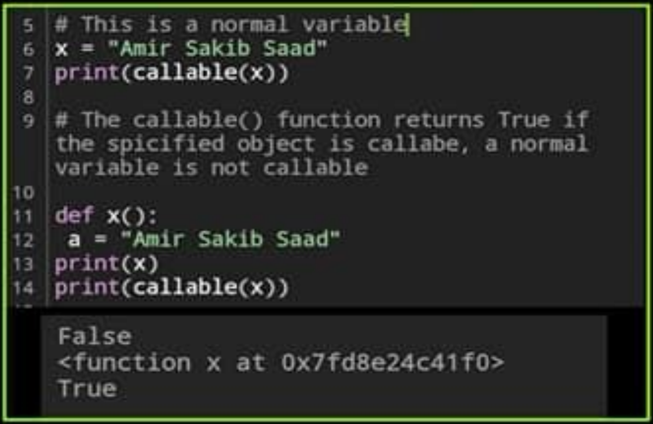
complex( )



```python
# The complex() function returns a complex number by specifing a real number and a imaginary number.

# complex(real,imaginary)

x = complex(12,3)
print(x)
```

```
(12+3j)
```

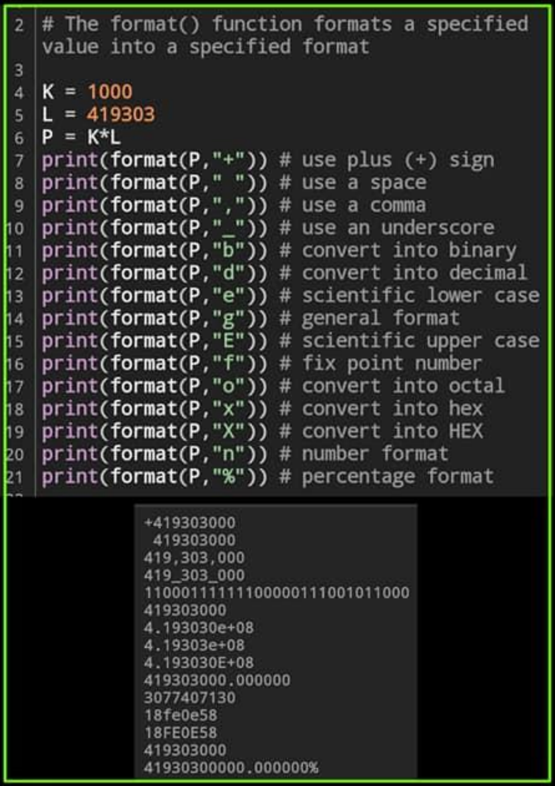
callable( )



```python
# This is a normal variable
x = "Amir Sakib Saad"
print(callable(x))

# The callable() function returns True if the spicified object is callabe, a normal variable is not callable

def x():
 a = "Amir Sakib Saad"
print(x)
print(callable(x))
```

```
False
<function x at 0x7fd8e24c41f0>
True
```

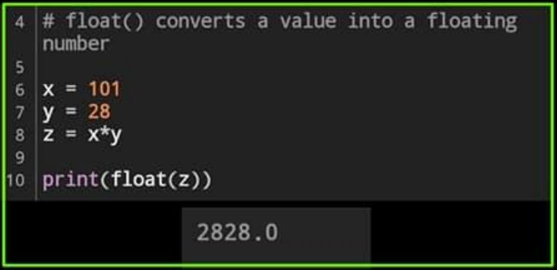
format( )



```python
# The format() function formats a specified
value into a specified format

K = 1000
L = 419303
P = K*L
print(format(P,"+")) # use plus (+) sign
print(format(P," ")) # use a space
print(format(P,",")) # use a comma
print(format(P,"_")) # use an underscore
print(format(P,"b")) # convert into binary
print(format(P,"d")) # convert into decimal
print(format(P,"e")) # scientific lower case
print(format(P,"g")) # general format
print(format(P,"E")) # scientific upper case
print(format(P,"f")) # fix point number
print(format(P,"o")) # convert into octal
print(format(P,"x")) # convert into hex
print(format(P,"X")) # convert into HEX
print(format(P,"n")) # number format
print(format(P,"%")) # percentage format
```

```
+419303000
 419303000
419,303,000
419_303_000
11000111111100000111001011000
419303000
4.193030e+08
4.19303e+08
4.193030E+08
419303000.000000
3077407130
18fe0e58
18FE0E58
419303000
41930300000.000000%
```

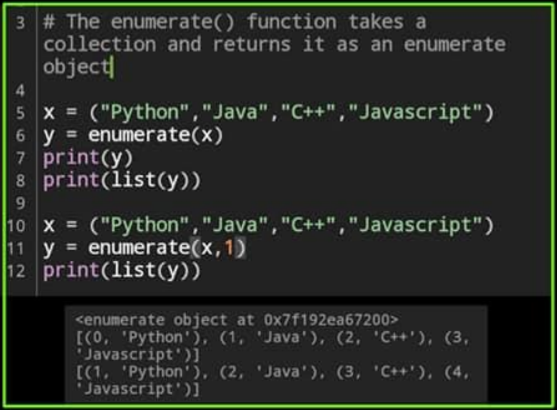
float( )



```python
# float() converts a value into a floating
number

x = 101
y = 28
z = x*y

print(float(z))
```

```
2828.0
```

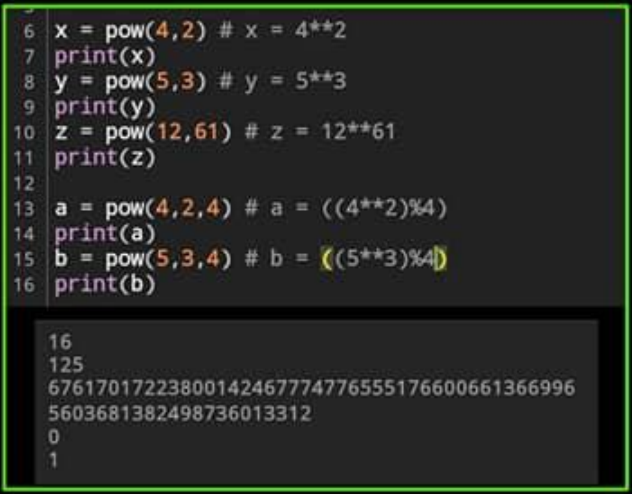
enumerate( )



```python
# The enumerate() function takes a
collection and returns it as an enumerate
object

x = ("Python","Java","C++","Javascript")
y = enumerate(x)
print(y)
print(list(y))

x = ("Python","Java","C++","Javascript")
y = enumerate(x,1)
print(list(y))
```

```
<enumerate object at 0x7f192ea67200>
[(0, 'Python'), (1, 'Java'), (2, 'C++'), (3,
'Javascript')]
[(1, 'Python'), (2, 'Java'), (3, 'C++'), (4,
'Javascript')]
```

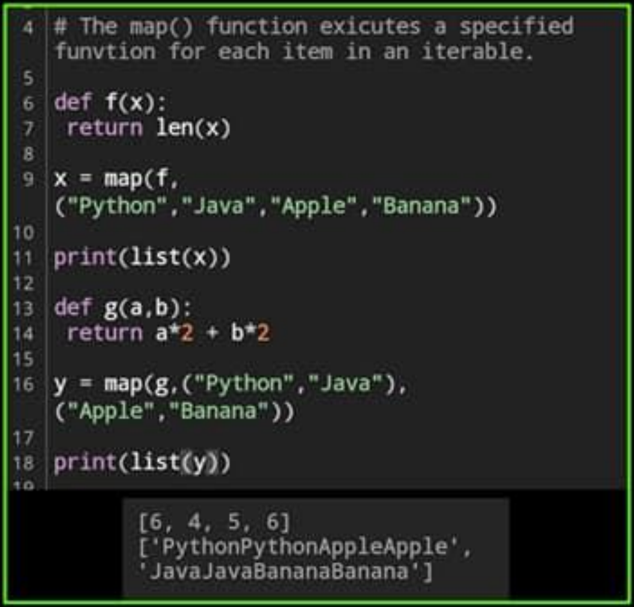
pow( )



```python
x = pow(4,2) # x = 4**2
print(x)
y = pow(5,3) # y = 5**3
print(y)
z = pow(12,61) # z = 12**61
print(z)

a = pow(4,2,4) # a = ((4**2)%4)
print(a)
b = pow(5,3,4) # b = ((5**3)%4)
print(b)
```

```
16
125
676170172238001424677747765551766006613669965
6036813824987360133 12
0
1
```

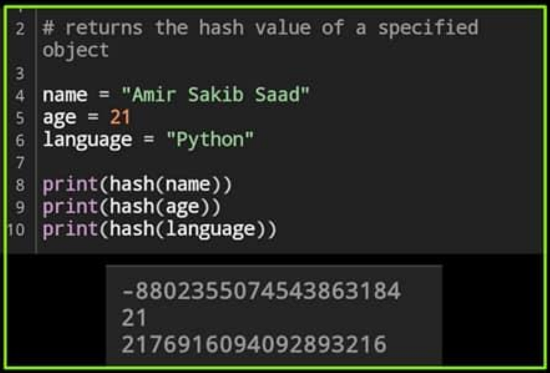
map( )



```python
# The map() function exicutes a specified
funvtion for each item in an iterable.

def f(x):
 return len(x)

x = map(f,
("Python","Java","Apple","Banana"))

print(list(x))

def g(a,b):
 return a*2 + b*2

y = map(g,("Python","Java"),
("Apple","Banana"))

print(list(y))
```

```
[6, 4, 5, 6]
['PythonPythonAppleApple',
'JavaJavaBananaBanana']
```

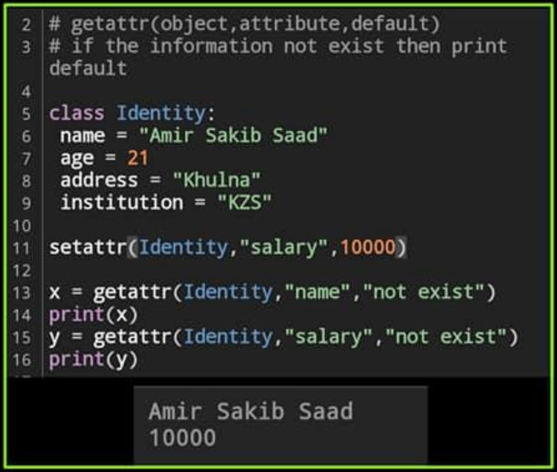
hash( )



```python
# returns the hash value of a specified
object

name = "Amir Sakib Saad"
age = 21
language = "Python"

print(hash(name))
print(hash(age))
print(hash(language))
```

```
-8802355074543863184
21
2176916094092893216
```

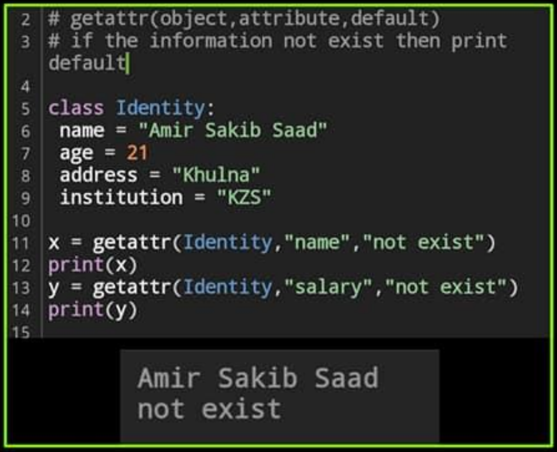
setattr( )



```python
# getattr(object,attribute,default)
# if the information not exist then print
default

class Identity:
 name = "Amir Sakib Saad"
 age = 21
 address = "Khulna"
 institution = "KZS"

setattr(Identity,"salary",10000)

x = getattr(Identity,"name","not exist")
print(x)
y = getattr(Identity,"salary","not exist")
print(y)
```

```
Amir Sakib Saad
10000
```

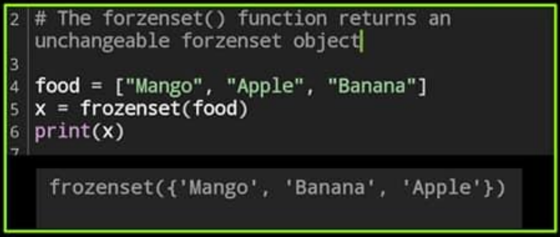
getattr( )



```python
# getattr(object,attribute,default)
# if the information not exist then print
default

class Identity:
 name = "Amir Sakib Saad"
 age = 21
 address = "Khulna"
 institution = "KZS"

x = getattr(Identity,"name","not exist")
print(x)
y = getattr(Identity,"salary","not exist")
print(y)
```

```
Amir Sakib Saad
not exist
```

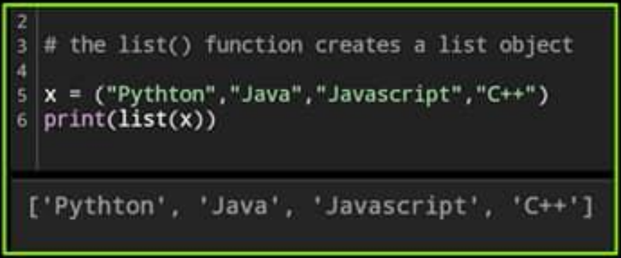
frozenset( )



```python
# The forzenset() function returns an
unchangeable forzenset object

food = ["Mango", "Apple", "Banana"]
x = frozenset(food)
print(x)
```

```
frozenset({'Mango', 'Banana', 'Apple'})
```

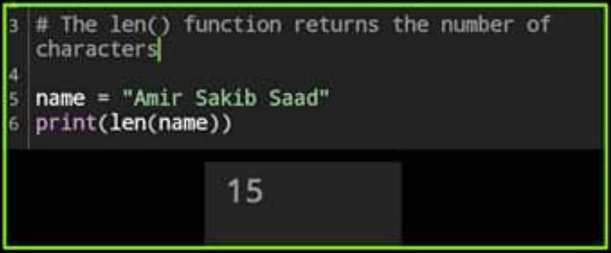
list( )



```python
# the list() function creates a list object

x = ("Pythton","Java","Javascript","C++")
print(list(x))
```

```
['Pythton', 'Java', 'Javascript', 'C++']
```

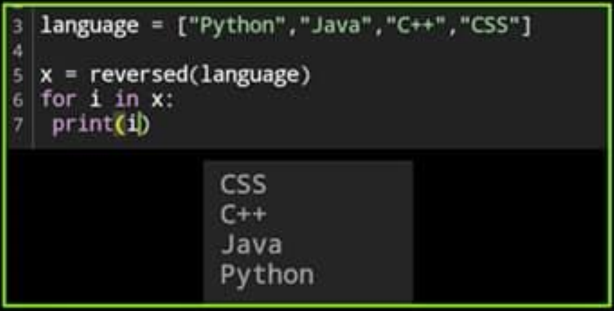
len( )



```python
# The len() function returns the number of characters

name = "Amir Sakib Saad"
print(len(name))
```

```
15
```

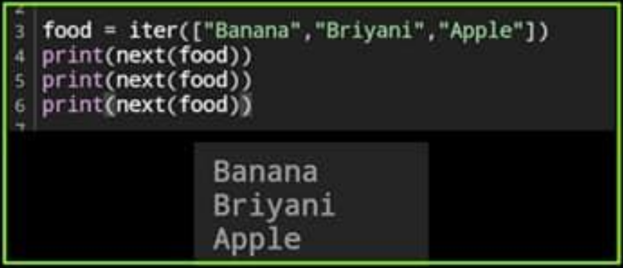
reversed( )



```python
language = ["Python","Java","C++","CSS"]

x = reversed(language)
for i in x:
 print(i)
```

```
CSS
C++
Java
Python
```

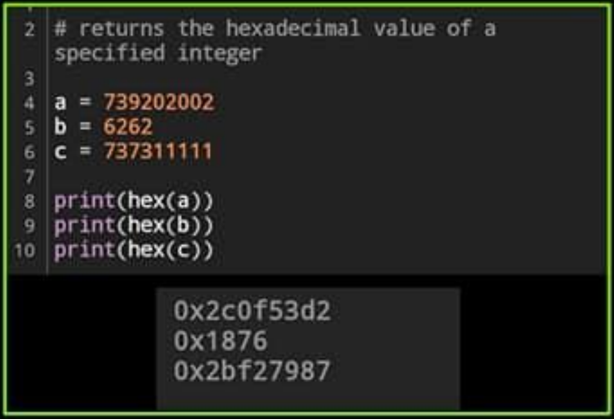
iter( )



```python
food = iter(["Banana","Briyani","Apple"])
print(next(food))
print(next(food))
print(next(food))
```

```
Banana
Briyani
Apple
```

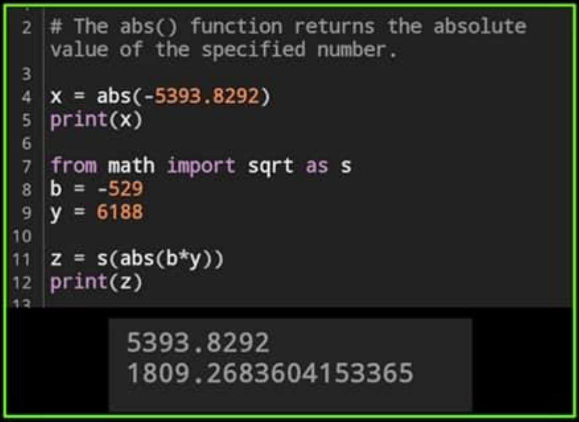
hex( )



```python
# returns the hexadecimal value of a specified integer

a = 739202002
b = 6262
c = 737311111

print(hex(a))
print(hex(b))
print(hex(c))
```

```
0x2c0f53d2
0x1876
0x2bf27987
```

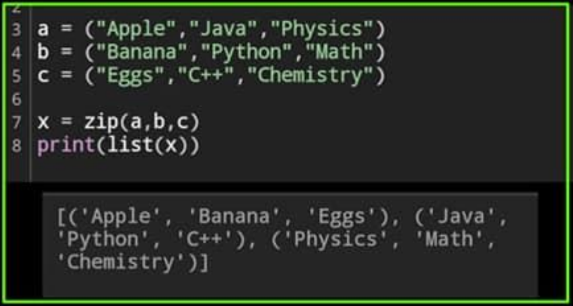
abs( )



```python
# The abs() function returns the absolute
value of the specified number.

x = abs(-5393.8292)
print(x)

from math import sqrt as s
b = -529
y = 6188

z = s(abs(b*y))
print(z)
```

```
5393.8292
1809.2683604153365
```

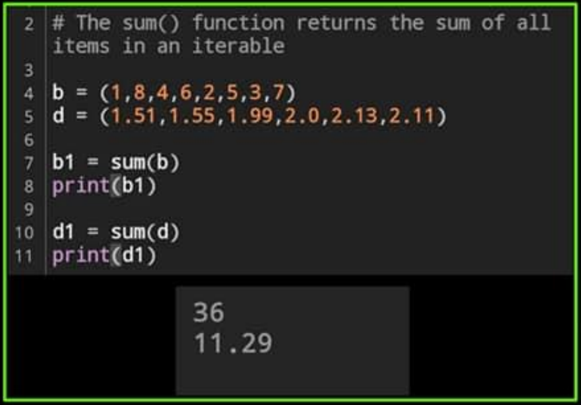
zip( )



```python
a = ("Apple","Java","Physics")
b = ("Banana","Python","Math")
c = ("Eggs","C++","Chemistry")

x = zip(a,b,c)
print(list(x))
```

```
[('Apple', 'Banana', 'Eggs'), ('Java',
'Python', 'C++'), ('Physics', 'Math',
'Chemistry')]
```

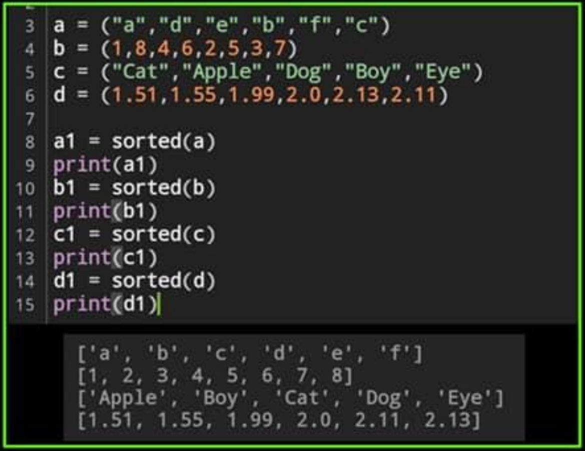
sum( )



```python
# The sum() function returns the sum of all
items in an iterable

b = (1,8,4,6,2,5,3,7)
d = (1.51,1.55,1.99,2.0,2.13,2.11)

b1 = sum(b)
print(b1)

d1 = sum(d)
print(d1)
```

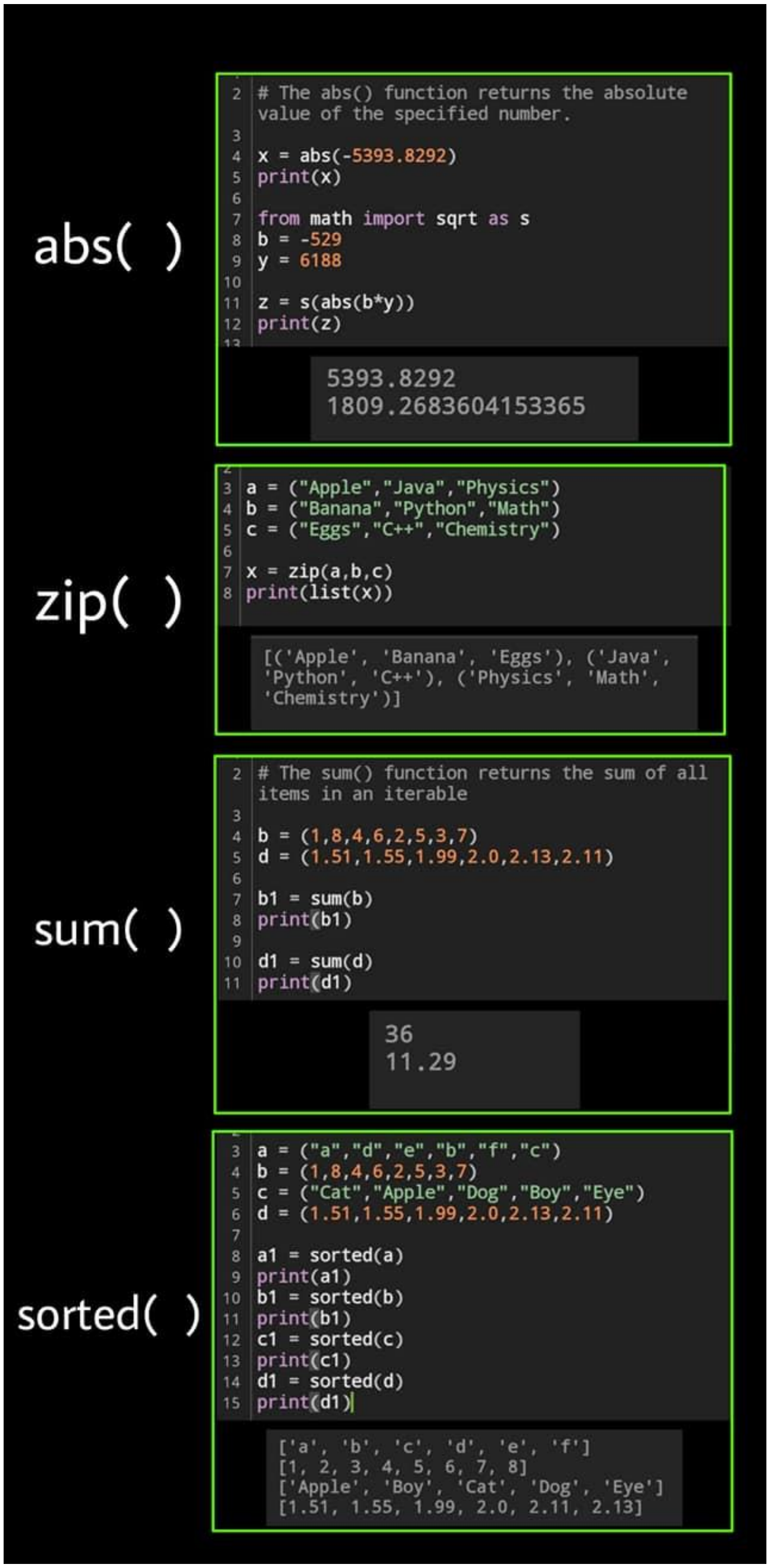
```
36
11.29
```

sorted( )



```python
a = ("a","d","e","b","f","c")
b = (1,8,4,6,2,5,3,7)
c = ("Cat","Apple","Dog","Boy","Eye")
d = (1.51,1.55,1.99,2.0,2.13,2.11)

a1 = sorted(a)
print(a1)
b1 = sorted(b)
print(b1)
c1 = sorted(c)
print(c1)
d1 = sorted(d)
print(d1)
```

```
['a', 'b', 'c', 'd', 'e', 'f']
[1, 2, 3, 4, 5, 6, 7, 8]
['Apple', 'Boy', 'Cat', 'Dog', 'Eye']
[1.51, 1.55, 1.99, 2.0, 2.11, 2.13]
```



abs( )

```python
# The abs() function returns the absolute
value of the specified number.

x = abs(-5393.8292)
print(x)

from math import sqrt as s
b = -529
y = 6188

z = s(abs(b*y))
print(z)
```

```
5393.8292
1809.2683604153365
```

zip( )

```python
a = ("Apple","Java","Physics")
b = ("Banana","Python","Math")
c = ("Eggs","C++","Chemistry")

x = zip(a,b,c)
print(list(x))
```

```
[('Apple', 'Banana', 'Eggs'), ('Java',
'Python', 'C++'), ('Physics', 'Math',
'Chemistry')]
```

sum( )

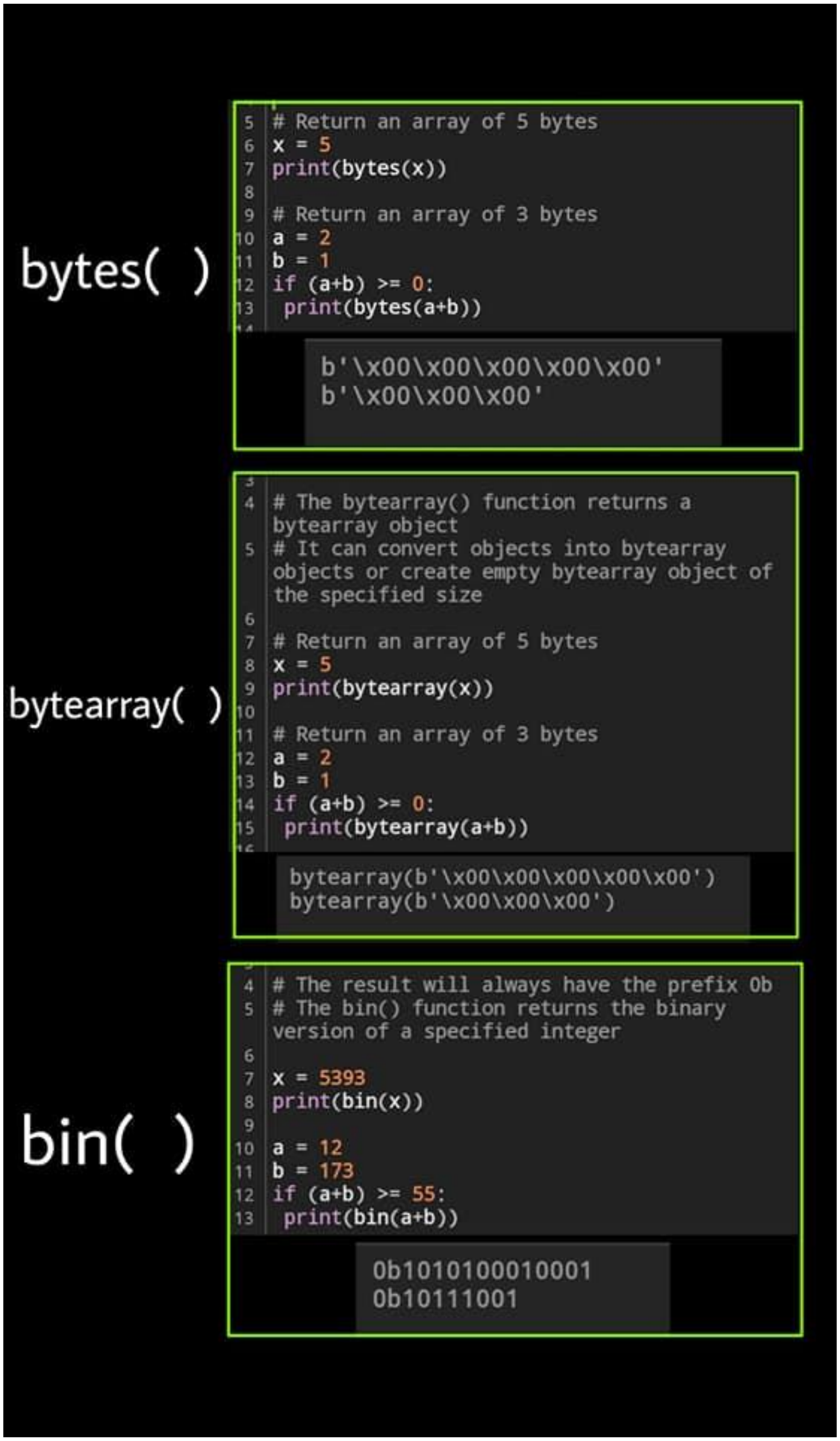
```python
# The sum() function returns the sum of all
items in an iterable

b = (1,8,4,6,2,5,3,7)
d = (1.51,1.55,1.99,2.0,2.13,2.11)

b1 = sum(b)
print(b1)

d1 = sum(d)
print(d1)
```

```
36
11.29
```

sorted( )

```python
a = ("a","d","e","b","f","c")
b = (1,8,4,6,2,5,3,7)
c = ("Cat","Apple","Dog","Boy","Eye")
d = (1.51,1.55,1.99,2.0,2.13,2.11)

a1 = sorted(a)
print(a1)
b1 = sorted(b)
print(b1)
c1 = sorted(c)
print(c1)
d1 = sorted(d)
print(d1)
```

```
['a', 'b', 'c', 'd', 'e', 'f']
[1, 2, 3, 4, 5, 6, 7, 8]
['Apple', 'Boy', 'Cat', 'Dog', 'Eye']
[1.51, 1.55, 1.99, 2.0, 2.11, 2.13]
```



bytes( )

```python
# Return an array of 5 bytes
x = 5
print(bytes(x))

# Return an array of 3 bytes
a = 2
b = 1
if (a+b) >= 0:
 print(bytes(a+b))
```

```
b'\x00\x00\x00\x00\x00'
b'\x00\x00\x00'
```

bytearray( )

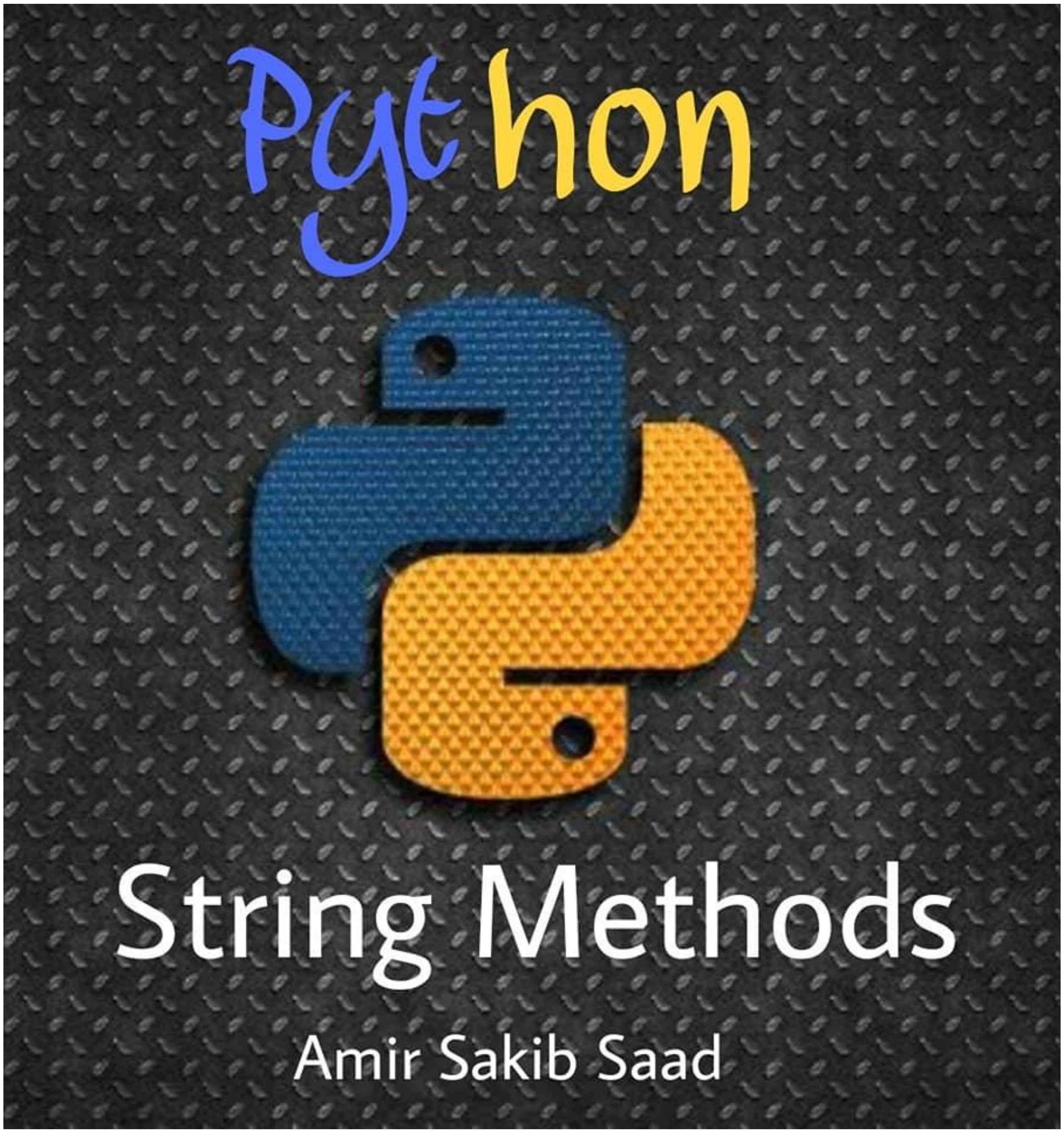
```python
# The bytearray() function returns a bytearray object
# It can convert objects into bytearray objects or create empty bytearray object of the specified size

# Return an array of 5 bytes
x = 5
print(bytearray(x))

# Return an array of 3 bytes
a = 2
b = 1
if (a+b) >= 0:
 print(bytearray(a+b))
```

```
bytearray(b'\x00\x00\x00\x00\x00')
bytearray(b'\x00\x00\x00')
```

bin( )

```python
# The result will always have the prefix 0b
# The bin() function returns the binary version of a specified integer

x = 5393
print(bin(x))

a = 12
b = 173
if (a+b) >= 55:
 print(bin(a+b))
```

```
0b1010100010001
0b10111001
```

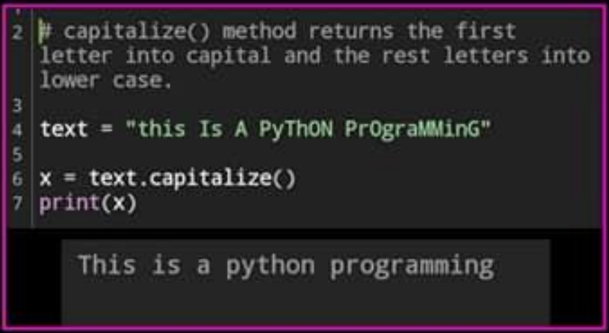


Python

String Methods

Amir Sakib Saad

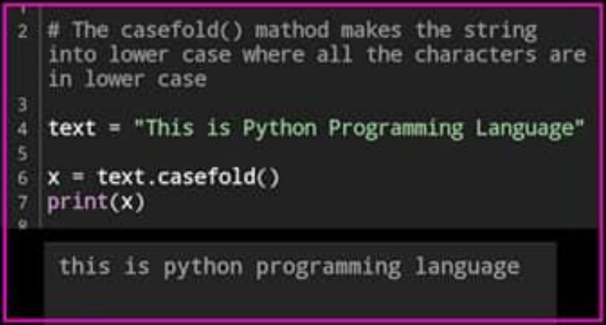
capitalize( )



```python
# capitalize() method returns the first letter into capital and the rest letters into lower case.

text = "this Is A PyThON PrOgraMMinG"

x = text.capitalize()
print(x)
```

```
This is a python programming
```

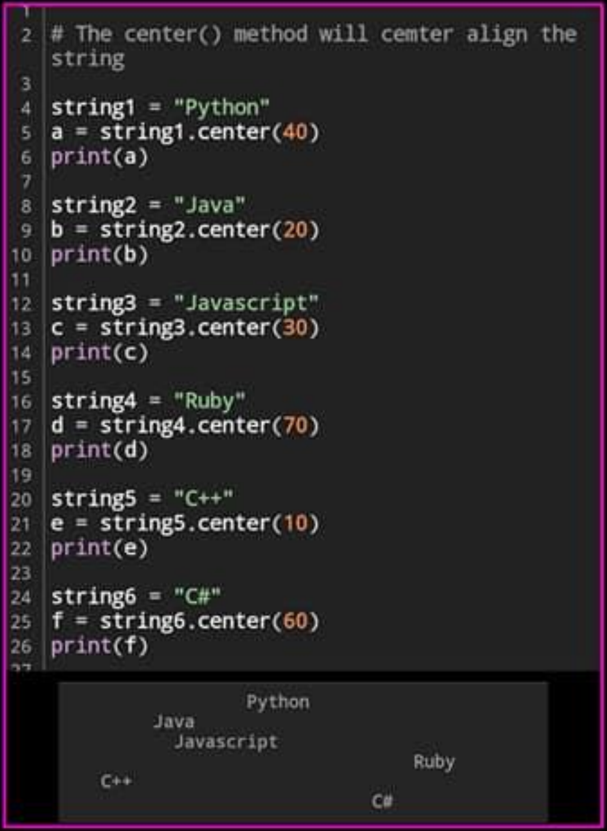
casefold( )



```python
# The casefold() mathod makes the string into lower case where all the characters are in lower case

text = "This is Python Programming Language"

x = text.casefold()
print(x)
```

```
this is python programming language
```

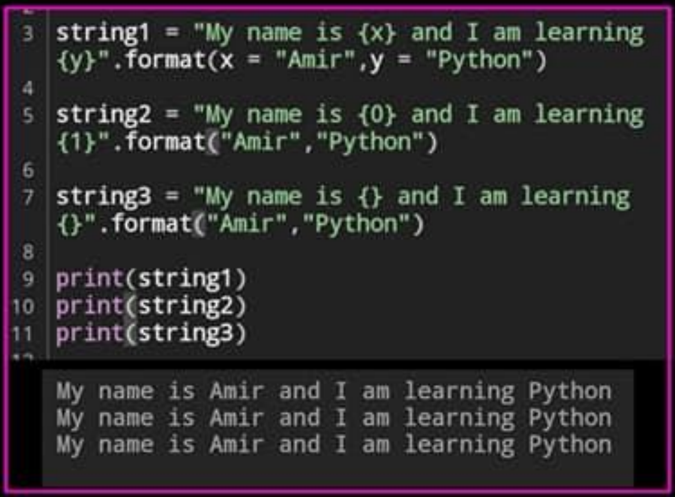
center( )



```python
# The center() method will cemter align the string

string1 = "Python"
a = string1.center(40)
print(a)

string2 = "Java"
b = string2.center(20)
print(b)

string3 = "Javascript"
c = string3.center(30)
print(c)

string4 = "Ruby"
d = string4.center(70)
print(d)

string5 = "C++"
e = string5.center(10)
print(e)

string6 = "C#"
f = string6.center(60)
print(f)
```

```
                 Python
        Java
          Javascript
                                 Ruby
   C++
                             C#
```

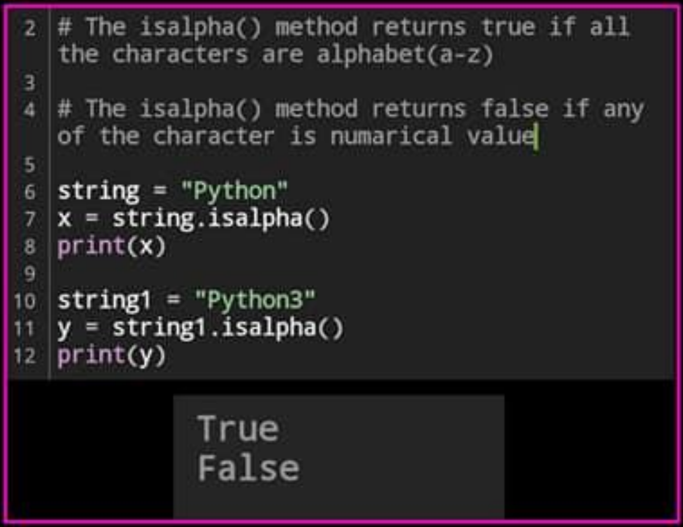
format( )



```python
string1 = "My name is {x} and I am learning {y}".format(x = "Amir",y = "Python")

string2 = "My name is {0} and I am learning {1}".format("Amir","Python")

string3 = "My name is {} and I am learning {}".format("Amir","Python")

print(string1)
print(string2)
print(string3)
```

```
My name is Amir and I am learning Python
My name is Amir and I am learning Python
My name is Amir and I am learning Python
```

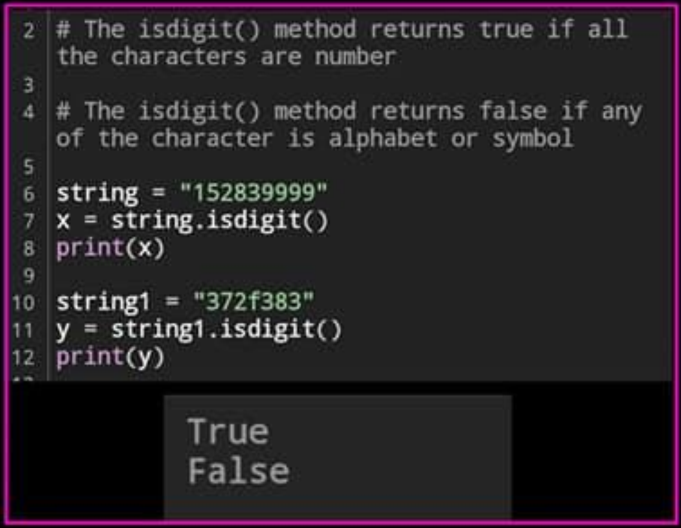
isalpha( )



```python
# The isalpha() method returns true if all the characters are alphabet(a-z)

# The isalpha() method returns false if any of the character is numarical value

string = "Python"
x = string.isalpha()
print(x)

string1 = "Python3"
y = string1.isalpha()
print(y)
```

```
True
False
```

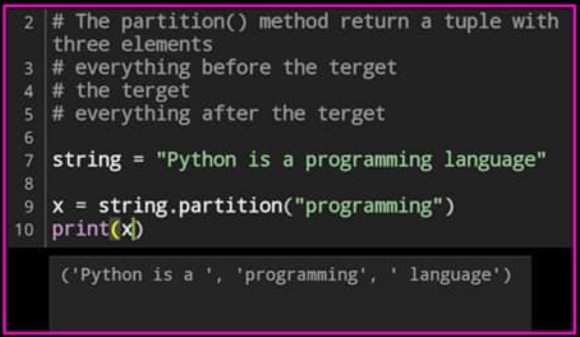
isdigit( )



```python
# The isdigit() method returns true if all the characters are number

# The isdigit() method returns false if any of the character is alphabet or symbol

string = "152839999"
x = string.isdigit()
print(x)

string1 = "372f383"
y = string1.isdigit()
print(y)
```

```
True
False
```

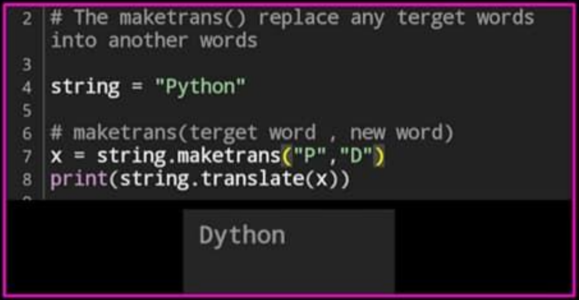
partition( )



```python
# The partition() method return a tuple with three elements
# everything before the terget
# the terget
# everything after the terget

string = "Python is a programming language"

x = string.partition("programming")
print(x)
```

```
('Python is a ', 'programming', ' language')
```

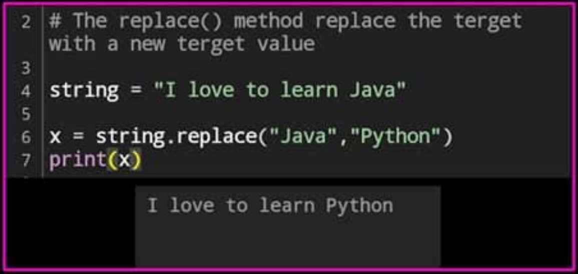
maketrans( )



```python
# The maketrans() replace any terget words into another words

string = "Python"

# maketrans(terget word , new word)
x = string.maketrans("P","D")
print(string.translate(x))
```

```
Dython
```

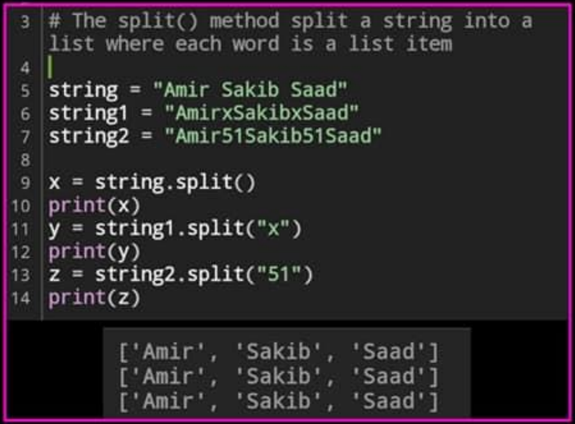
replace( )



```python
# The replace() method replace the terget with a new terget value

string = "I love to learn Java"

x = string.replace("Java","Python")
print(x)
```

```
I love to learn Python
```

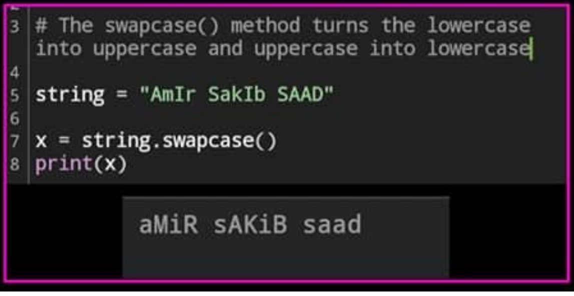
split( )



```python
# The split() method split a string into a list where each word is a list item

string = "Amir Sakib Saad"
string1 = "AmirxSakibxSaad"
string2 = "Amir51Sakib51Saad"

x = string.split()
print(x)
y = string1.split("x")
print(y)
z = string2.split("51")
print(z)
```

```
['Amir', 'Sakib', 'Saad']
['Amir', 'Sakib', 'Saad']
['Amir', 'Sakib', 'Saad']
```

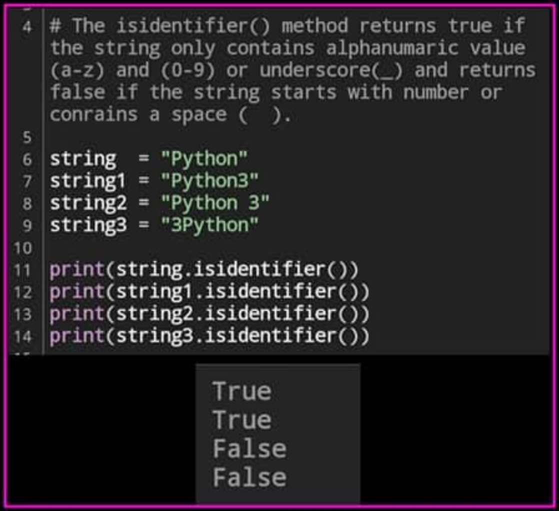
swapcase( )



```python
# The swapcase() method turns the lowercase into uppercase and uppercase into lowercase

string = "AmIr SakIb SAAD"

x = string.swapcase()
print(x)
```

```
aMiR sAKiB saad
```

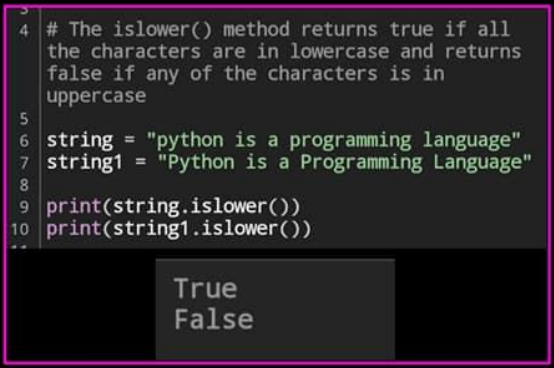
isidentifier( )



```python
# The isidentifier() method returns true if the string only contains alphanumaric value (a-z) and (0-9) or underscore(_) and returns false if the string starts with number or conrains a space (  ).

string  = "Python"
string1 = "Python3"
string2 = "Python 3"
string3 = "3Python"

print(string.isidentifier())
print(string1.isidentifier())
print(string2.isidentifier())
print(string3.isidentifier())
```

```
True
True
False
False
```

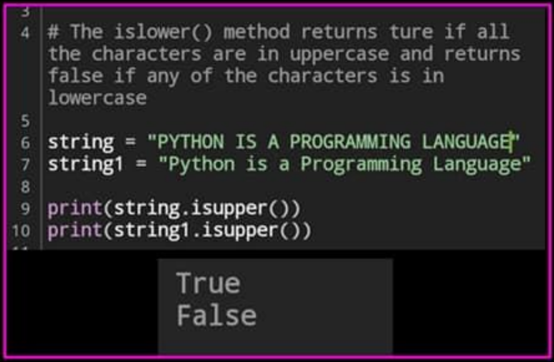
islower( )



```python
# The islower() method returns true if all the characters are in lowercase and returns false if any of the characters is in uppercase

string = "python is a programming language"
string1 = "Python is a Programming Language"

print(string.islower())
print(string1.islower())
```

```
True
False
```

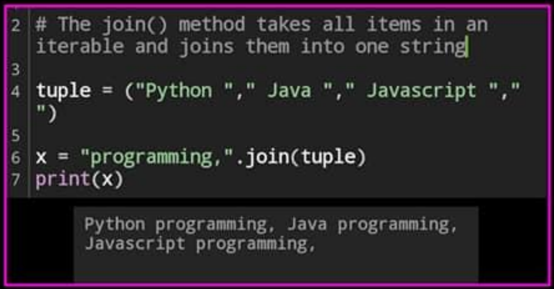
isupper( )



```python
# The islower() method returns ture if all the characters are in uppercase and returns false if any of the characters is in lowercase

string = "PYTHON IS A PROGRAMMING LANGUAGE"
string1 = "Python is a Programming Language"

print(string.isupper())
print(string1.isupper())
```

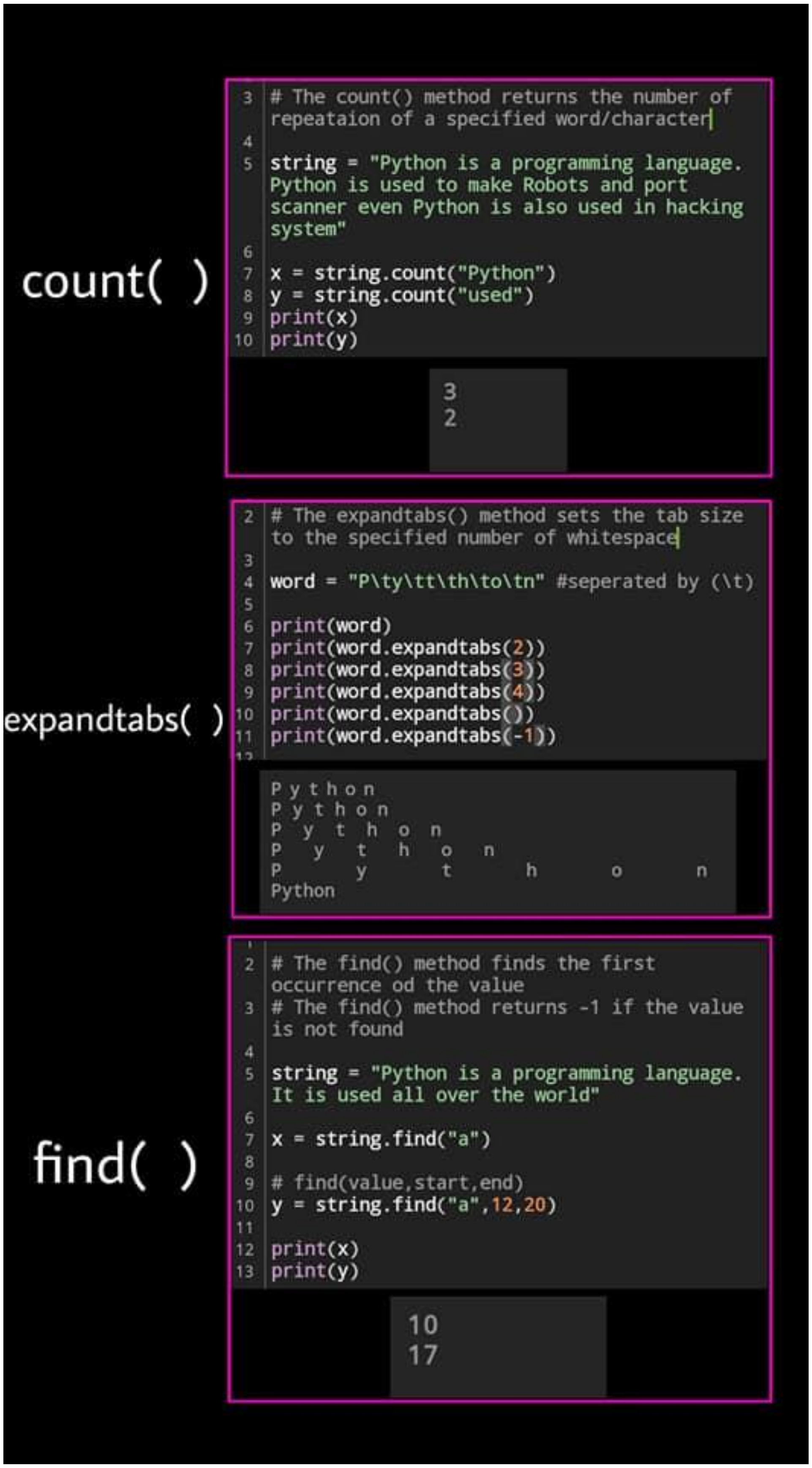
```
True
False
```

join( )



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

tuple = ("Python "," Java "," Javascript ","
")

x = "programming,".join(tuple)
print(x)
```

```
Python programming, Java programming,
Javascript programming,
```



count( )

```python
# The count() method returns the number of repeataion of a specified word/character

string = "Python is a programming language. Python is used to make Robots and port scanner even Python is also used in hacking system"

x = string.count("Python")
y = string.count("used")
print(x)
print(y)
```

```
3
2
```

expandtabs( )

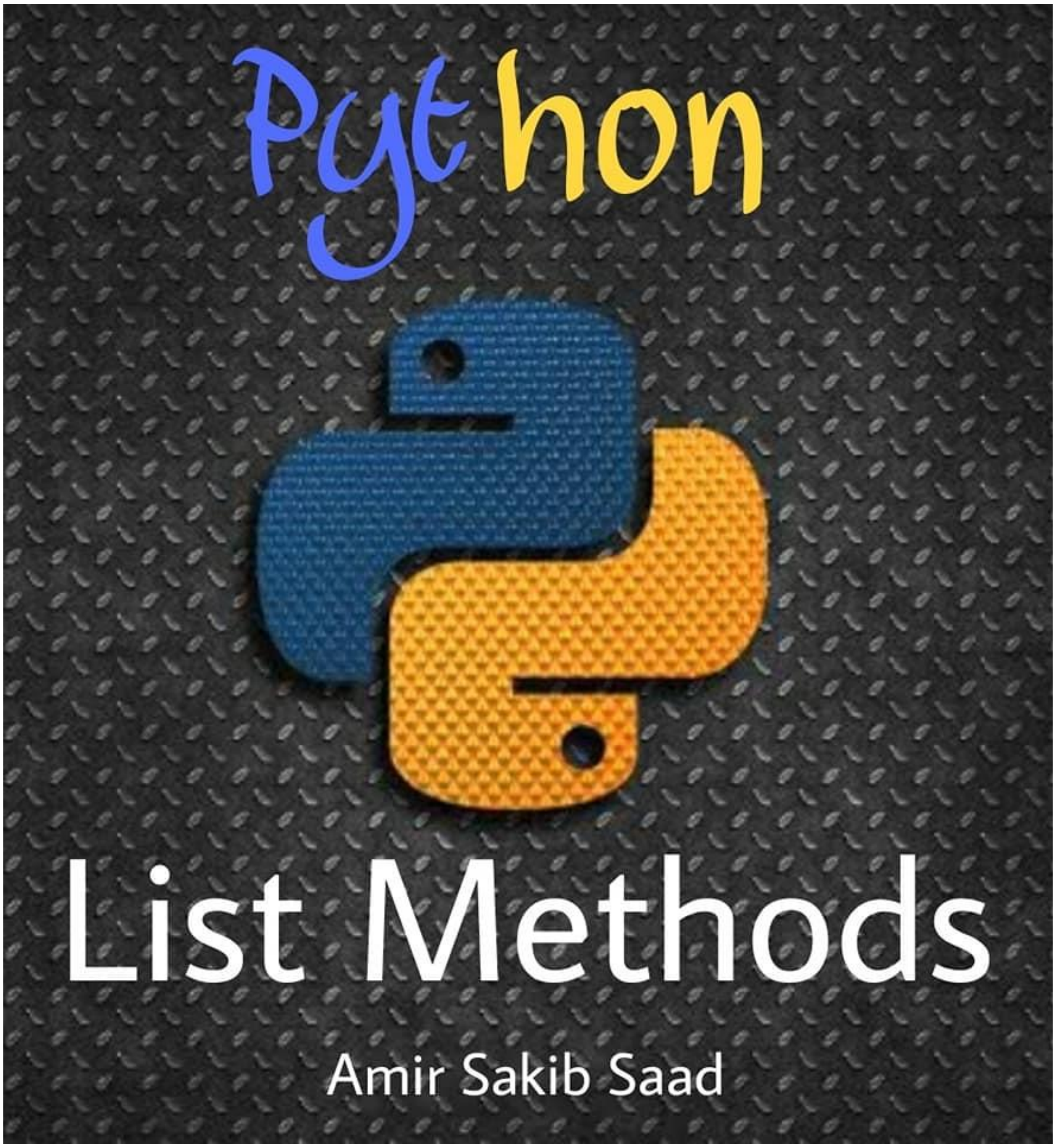
```python
# The expandtabs() method sets the tab size to the specified number of whitespace

word = "P\ty\tt\th\to\tn" #seperated by (\t)

print(word)
print(word.expandtabs(2))
print(word.expandtabs(3))
print(word.expandtabs(4))
print(word.expandtabs())
print(word.expandtabs(-1))
```

```
P y t h o n
P y t h o n
P  y  t  h  o  n
P   y   t   h   o   n
P       y       t       h       o       n
Python
```

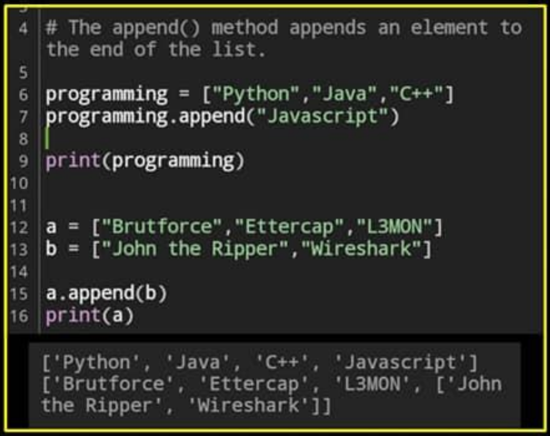
find( )

```python
# The find() method finds the first occurrence od the value
# The find() method returns -1 if the value is not found

string = "Python is a programming language. It is used all over the world"

x = string.find("a")

# find(value,start,end)
y = string.find("a",12,20)

print(x)
print(y)
```

```
10
17
```


Python
List Methods
Amir Sakib Saad

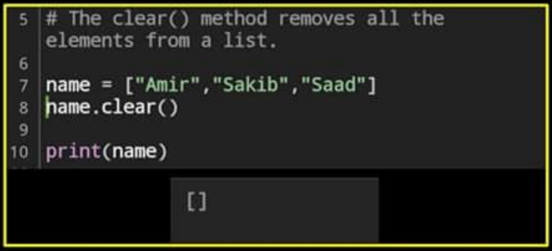
append( )



```python
# The append() method appends an element to the end of the list.

programming = ["Python","Java","C++"]
programming.append("Javascript")

print(programming)


a = ["Brutforce","Ettercap","L3MON"]
b = ["John the Ripper","Wireshark"]

a.append(b)
print(a)
```

```
['Python', 'Java', 'C++', 'Javascript']
['Brutforce', 'Ettercap', 'L3MON', ['John
the Ripper', 'Wireshark']]
```

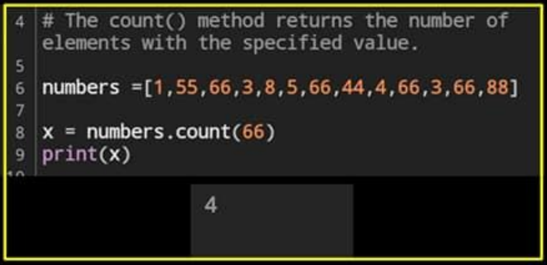
clear( )



```python
# The clear() method removes all the elements from a list.

name = ["Amir","Sakib","Saad"]
name.clear()

print(name)
```

```
[]
```

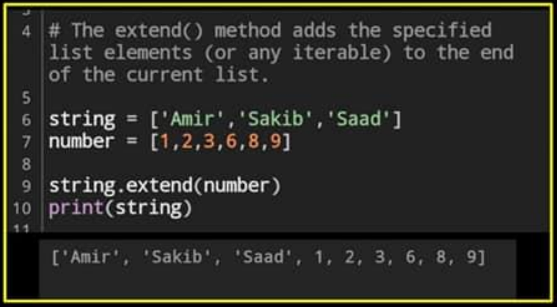
count( )



```python
# The count() method returns the number of elements with the specified value.

numbers =[1,55,66,3,8,5,66,44,4,66,3,66,88]

x = numbers.count(66)
print(x)
```

```
4
```

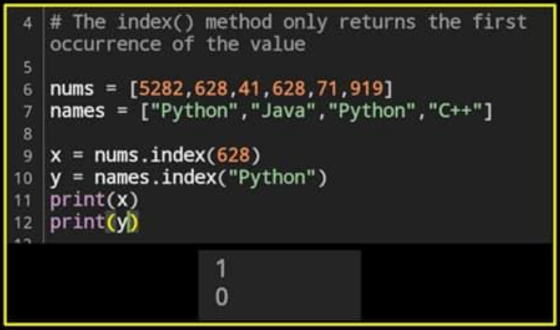
extend( )



```python
# The extend() method adds the specified list elements (or any iterable) to the end of the current list.

string = ['Amir','Sakib','Saad']
number = [1,2,3,6,8,9]

string.extend(number)
print(string)
```

```
['Amir', 'Sakib', 'Saad', 1, 2, 3, 6, 8, 9]
```

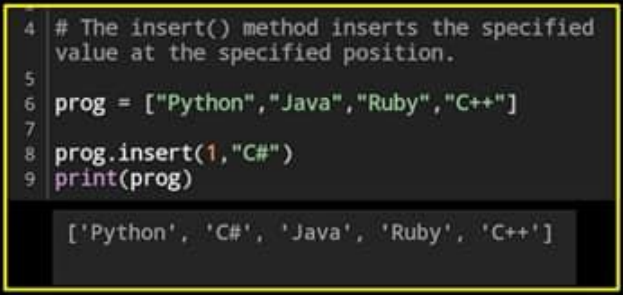
index( )



```python
# The index() method only returns the first occurrence of the value

nums = [5282,628,41,628,71,919]
names = ["Python","Java","Python","C++"]

x = nums.index(628)
y = names.index("Python")
print(x)
print(y)
```

```
1
0
```

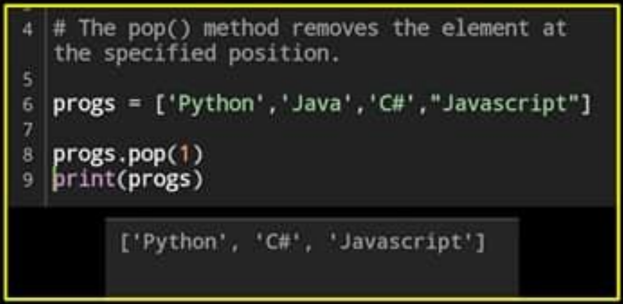
insert( )



```python
# The insert() method inserts the specified value at the specified position.

prog = ["Python","Java","Ruby","C++"]

prog.insert(1,"C#")
print(prog)
```

```
['Python', 'C#', 'Java', 'Ruby', 'C++']
```

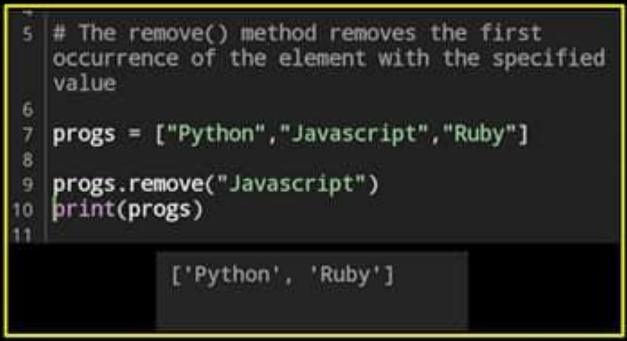
pop( )



```python
# The pop() method removes the element at the specified position.

progs = ['Python','Java','C#',"Javascript"]

progs.pop(1)
print(progs)
```

```
['Python', 'C#', 'Javascript']
```

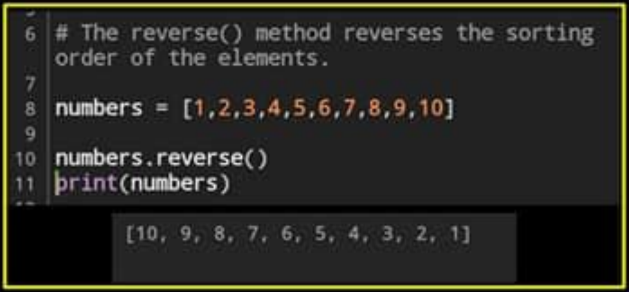
remove( )



```python
# The remove() method removes the first occurrence of the element with the specified value

progs = ["Python","Javascript","Ruby"]

progs.remove("Javascript")
print(progs)
```

```
['Python', 'Ruby']
```

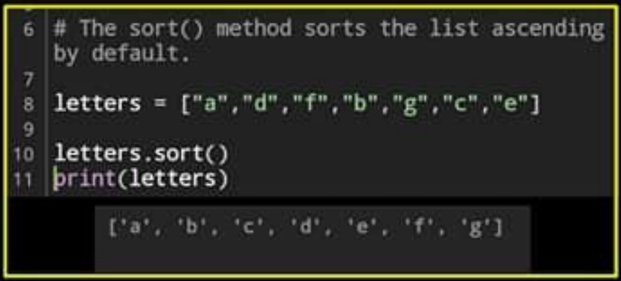
reverse( )



```python
# The reverse() method reverses the sorting order of the elements.

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

numbers.reverse()
print(numbers)
```

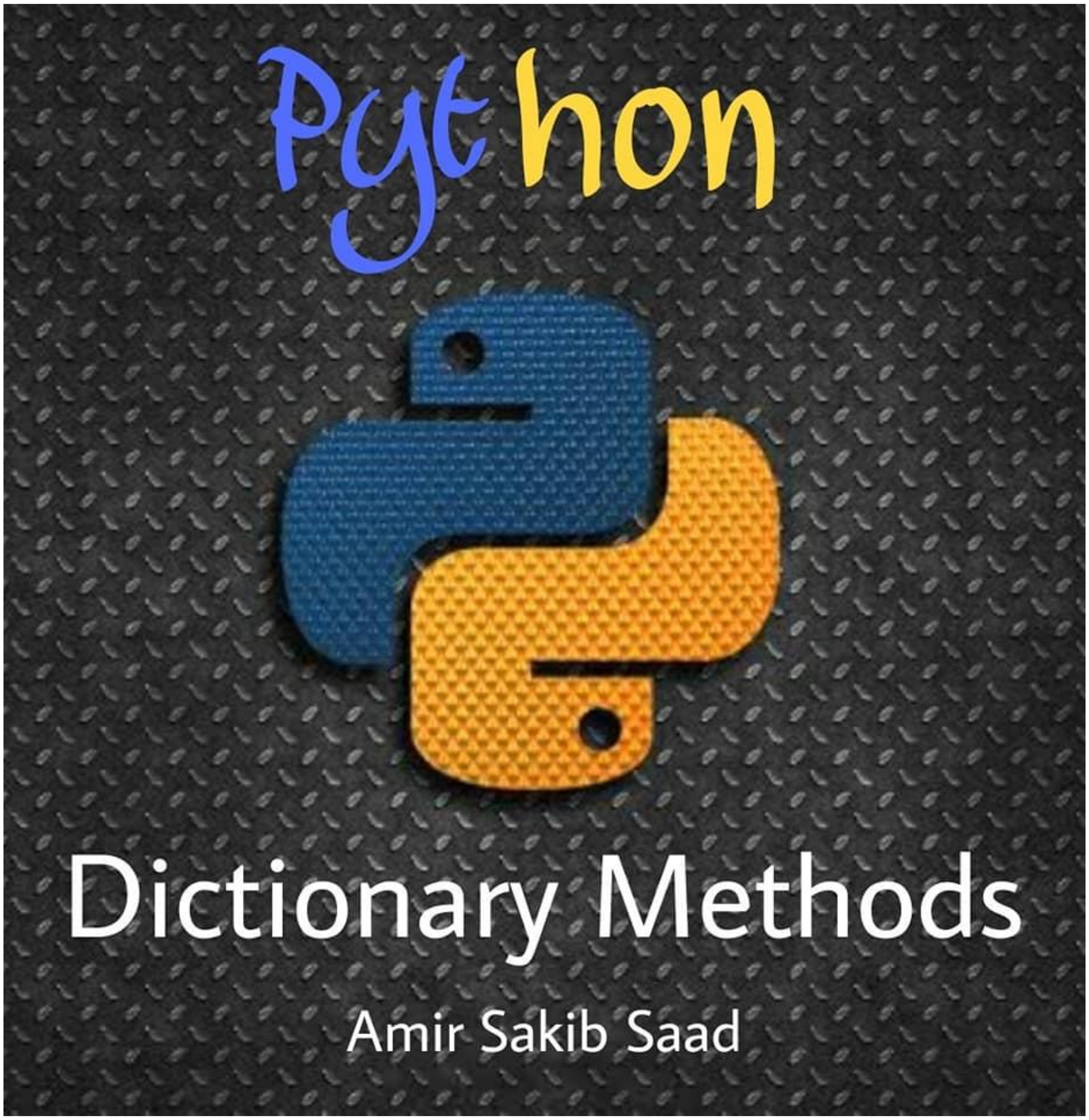
```
[10, 9, 8, 7, 6, 5, 4, 3, 2, 1]
```

sort( )

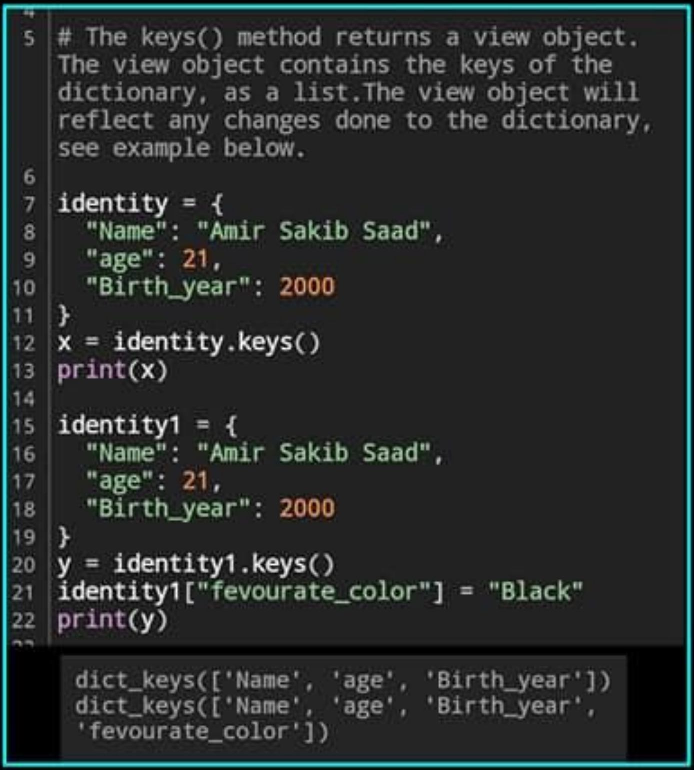


```python
# The sort() method sorts the list ascending by default.

letters = ["a","d","f","b","g","c","e"]

letters.sort()
print(letters)
```

```
['a', 'b', 'c', 'd', 'e', 'f', 'g']
```


Python
Dictionary Methods
Amir Sakib Saad

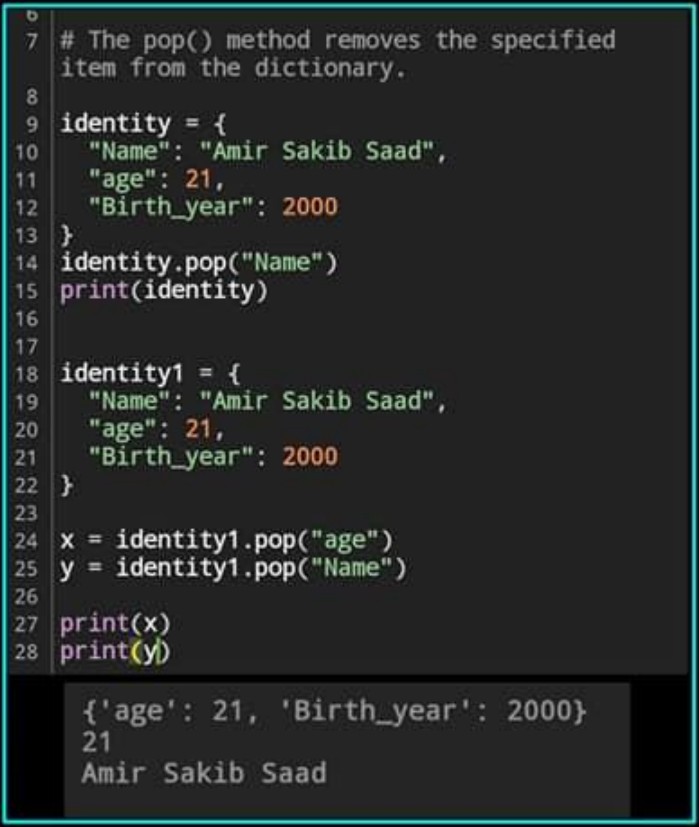
keys( )



```
# The keys() method returns a view object. The view object contains the keys of the dictionary, as a list.The view object will reflect any changes done to the dictionary, see example below.

identity = {
  "Name": "Amir Sakib Saad",
  "age": 21,
  "Birth_year": 2000
}
x = identity.keys()
print(x)

identity1 = {
  "Name": "Amir Sakib Saad",
  "age": 21,
  "Birth_year": 2000
}
y = identity1.keys()
identity1["fevourate_color"] = "Black"
print(y)
```

```
dict_keys(['Name', 'age', 'Birth_year'])
dict_keys(['Name', 'age', 'Birth_year',
'fevourate_color'])
```

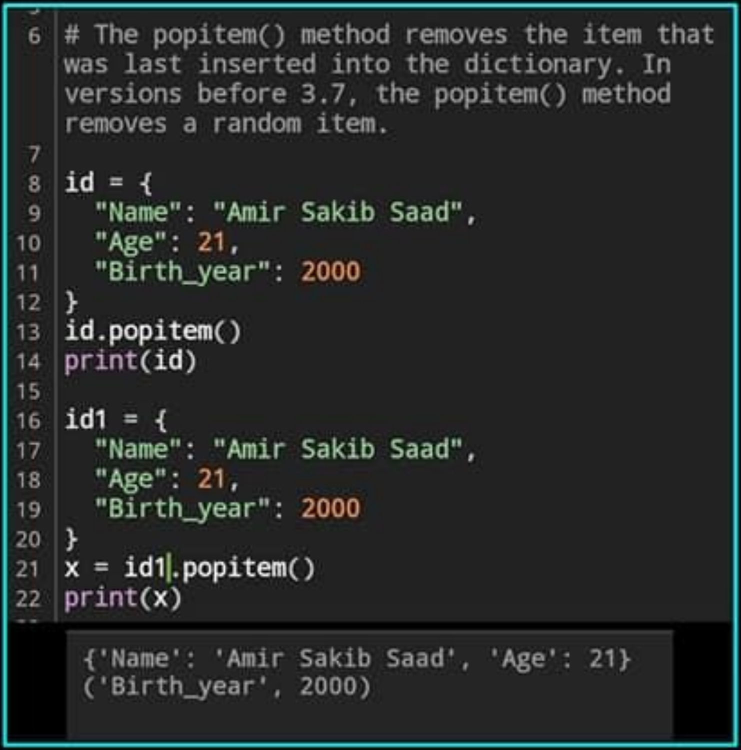
pop( )



```
# The pop() method removes the specified item from the dictionary.

identity = {
  "Name": "Amir Sakib Saad",
  "age": 21,
  "Birth_year": 2000
}
identity.pop("Name")
print(identity)


identity1 = {
  "Name": "Amir Sakib Saad",
  "age": 21,
  "Birth_year": 2000
}

x = identity1.pop("age")
y = identity1.pop("Name")

print(x)
print(y)
```

```
{'age': 21, 'Birth_year': 2000}
21
Amir Sakib Saad
```

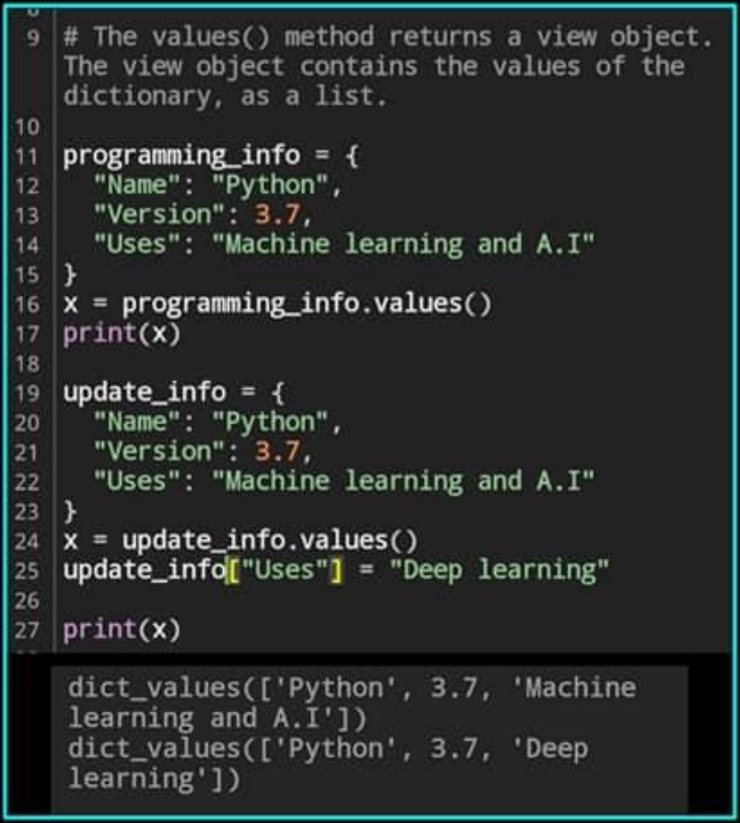
popitem( )



```python
# The popitem() method removes the item that was last inserted into the dictionary. In versions before 3.7, the popitem() method removes a random item.

id = {
  "Name": "Amir Sakib Saad",
  "Age": 21,
  "Birth_year": 2000
}
id.popitem()
print(id)

id1 = {
  "Name": "Amir Sakib Saad",
  "Age": 21,
  "Birth_year": 2000
}
x = id1.popitem()
print(x)
```

```
{'Name': 'Amir Sakib Saad', 'Age': 21}
('Birth_year', 2000)
```

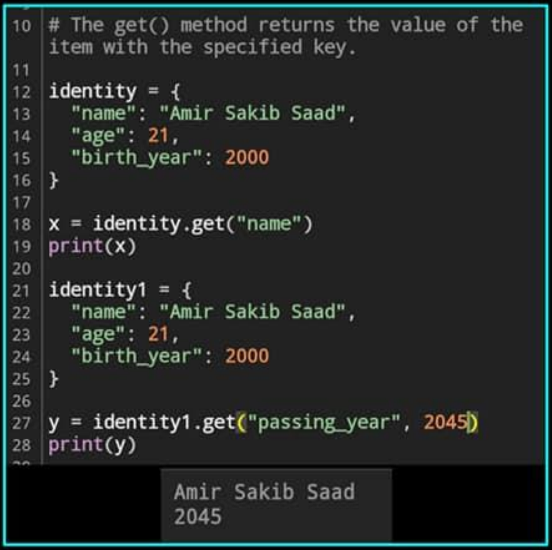
values( )



```python
# The values() method returns a view object. The view object contains the values of the dictionary, as a list.

programming_info = {
  "Name": "Python",
  "Version": 3.7,
  "Uses": "Machine learning and A.I"
}
x = programming_info.values()
print(x)

update_info = {
  "Name": "Python",
  "Version": 3.7,
  "Uses": "Machine learning and A.I"
}
x = update_info.values()
update_info["Uses"] = "Deep learning"

print(x)
```

```
dict_values(['Python', 3.7, 'Machine
learning and A.I'])
dict_values(['Python', 3.7, 'Deep
learning'])
```

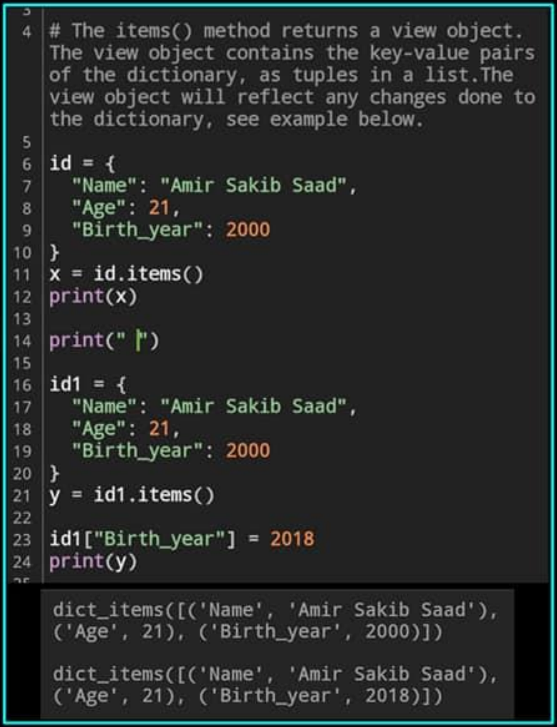
get( )



```python
# The get() method returns the value of the item with the specified key.

identity = {
  "name": "Amir Sakib Saad",
  "age": 21,
  "birth_year": 2000
}

x = identity.get("name")
print(x)

identity1 = {
  "name": "Amir Sakib Saad",
  "age": 21,
  "birth_year": 2000
}

y = identity1.get("passing_year", 2045)
print(y)
```

```
Amir Sakib Saad
2045
```

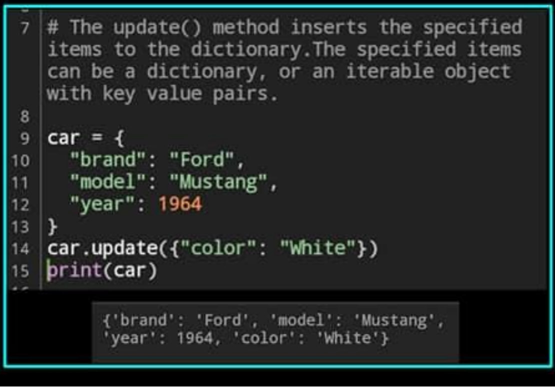
items( )



```python
# The items() method returns a view object. The view object contains the key-value pairs of the dictionary, as tuples in a list.The view object will reflect any changes done to the dictionary, see example below.

id = {
  "Name": "Amir Sakib Saad",
  "Age": 21,
  "Birth_year": 2000
}
x = id.items()
print(x)

print(" ")

id1 = {
  "Name": "Amir Sakib Saad",
  "Age": 21,
  "Birth_year": 2000
}
y = id1.items()

id1["Birth_year"] = 2018
print(y)
```

```
dict_items([('Name', 'Amir Sakib Saad'), ('Age', 21), ('Birth_year', 2000)])

dict_items([('Name', 'Amir Sakib Saad'), ('Age', 21), ('Birth_year', 2018)])
```

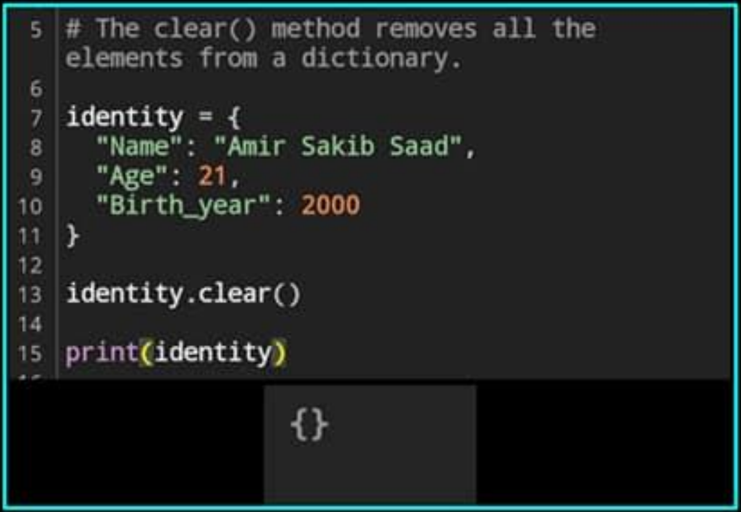
update( )



```python
# The update() method inserts the specified items to the dictionary.The specified items can be a dictionary, or an iterable object with key value pairs.

car = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
car.update({"color": "White"})
print(car)
```

```
{'brand': 'Ford', 'model': 'Mustang', 'year': 1964, 'color': 'White'}
```

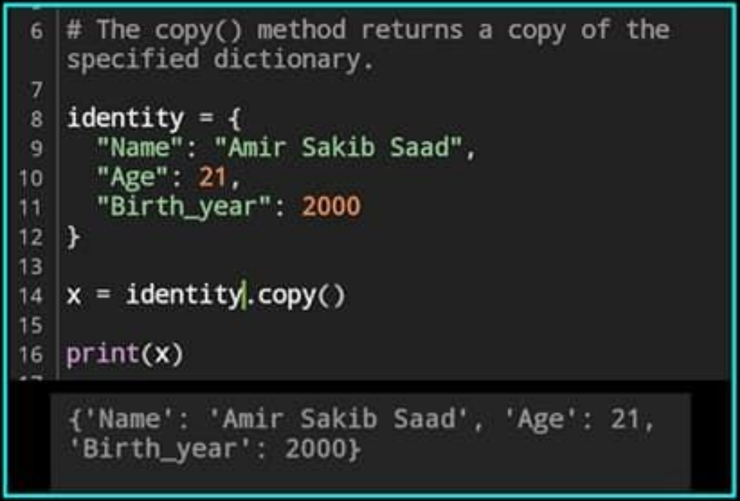
clear( )



```python
# The clear() method removes all the
elements from a dictionary.

identity = {
  "Name": "Amir Sakib Saad",
  "Age": 21,
  "Birth_year": 2000
}

identity.clear()

print(identity)
```

```
{}
```

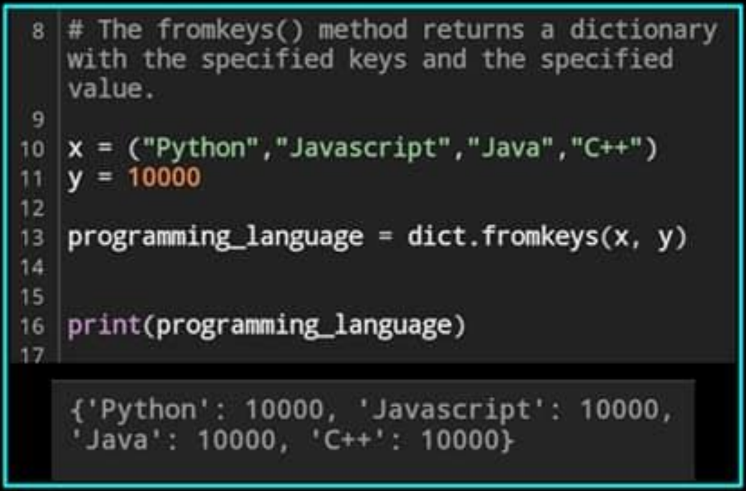
copy( )



```python
# The copy() method returns a copy of the
specified dictionary.

identity = {
  "Name": "Amir Sakib Saad",
  "Age": 21,
  "Birth_year": 2000
}

x = identity.copy()

print(x)
```

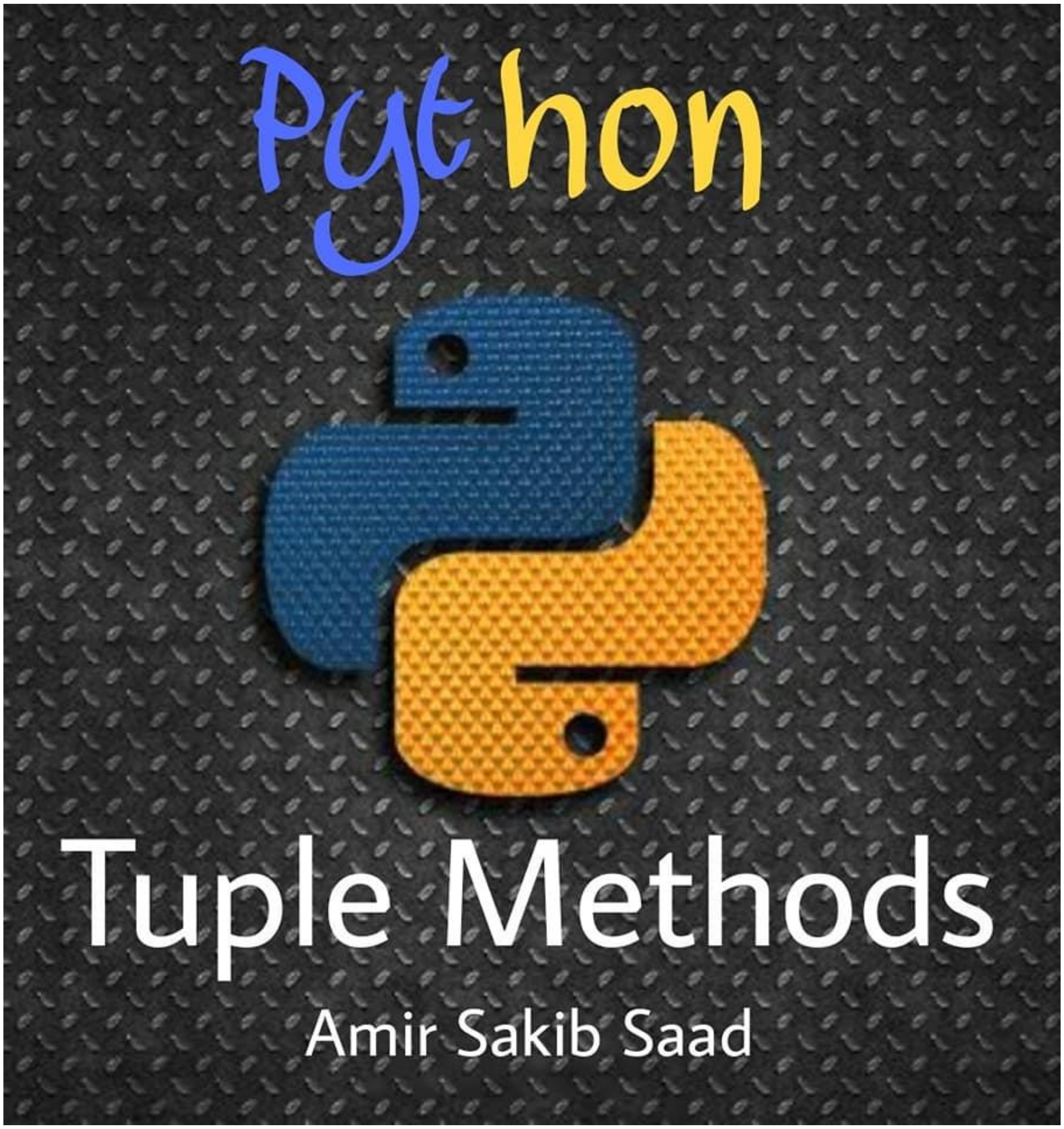
```
{'Name': 'Amir Sakib Saad', 'Age': 21,
'Birth_year': 2000}
```

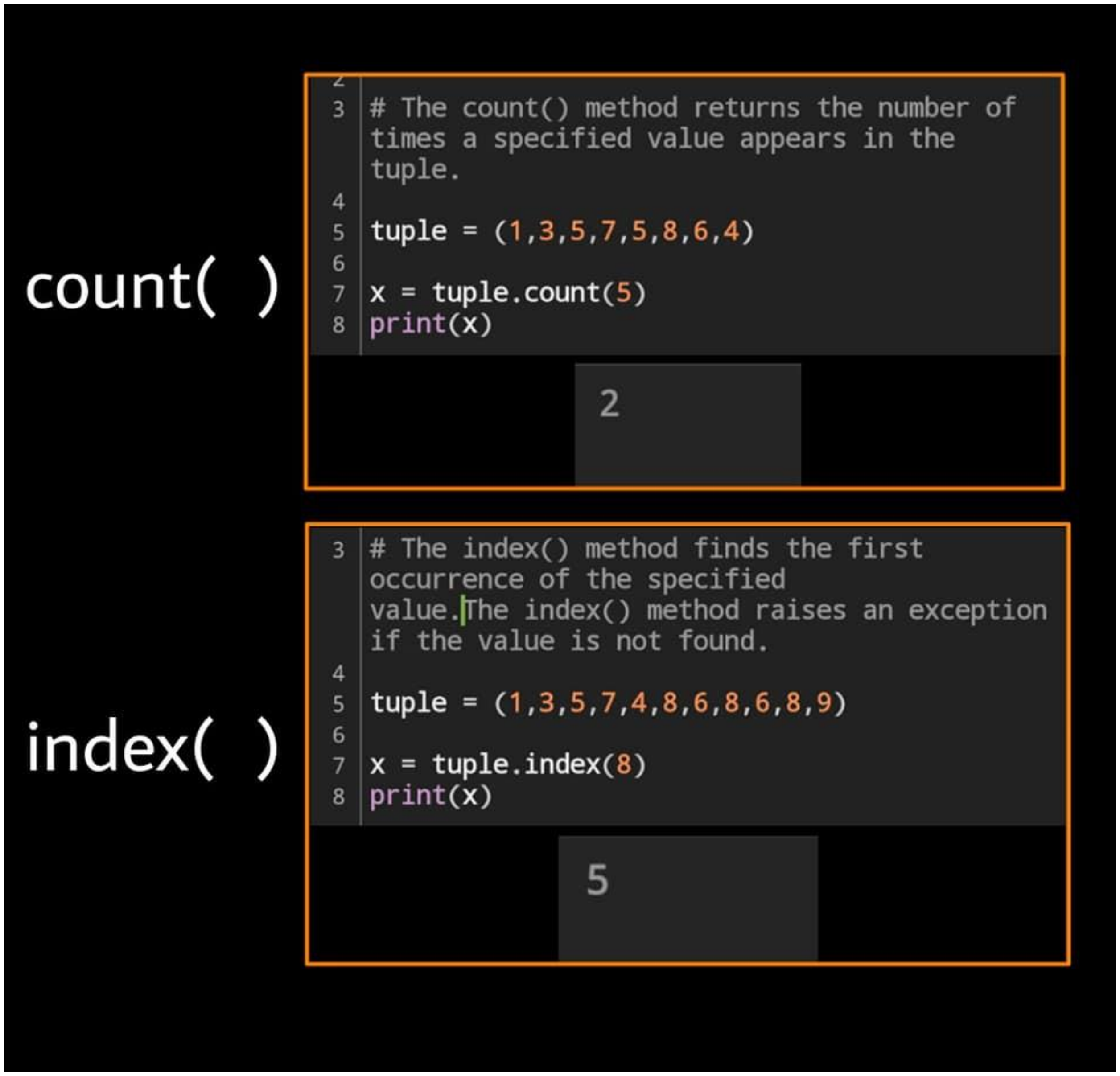
fromkeys( )



```python
# The fromkeys() method returns a dictionary
with the specified keys and the specified
value.

x = ("Python","Javascript","Java","C++")
y = 10000

programming_language = dict.fromkeys(x, y)


print(programming_language)
```

```
{'Python': 10000, 'Javascript': 10000,
'Java': 10000, 'C++': 10000}
```


Python
Tuple Methods
Amir Sakib Saad


count( )

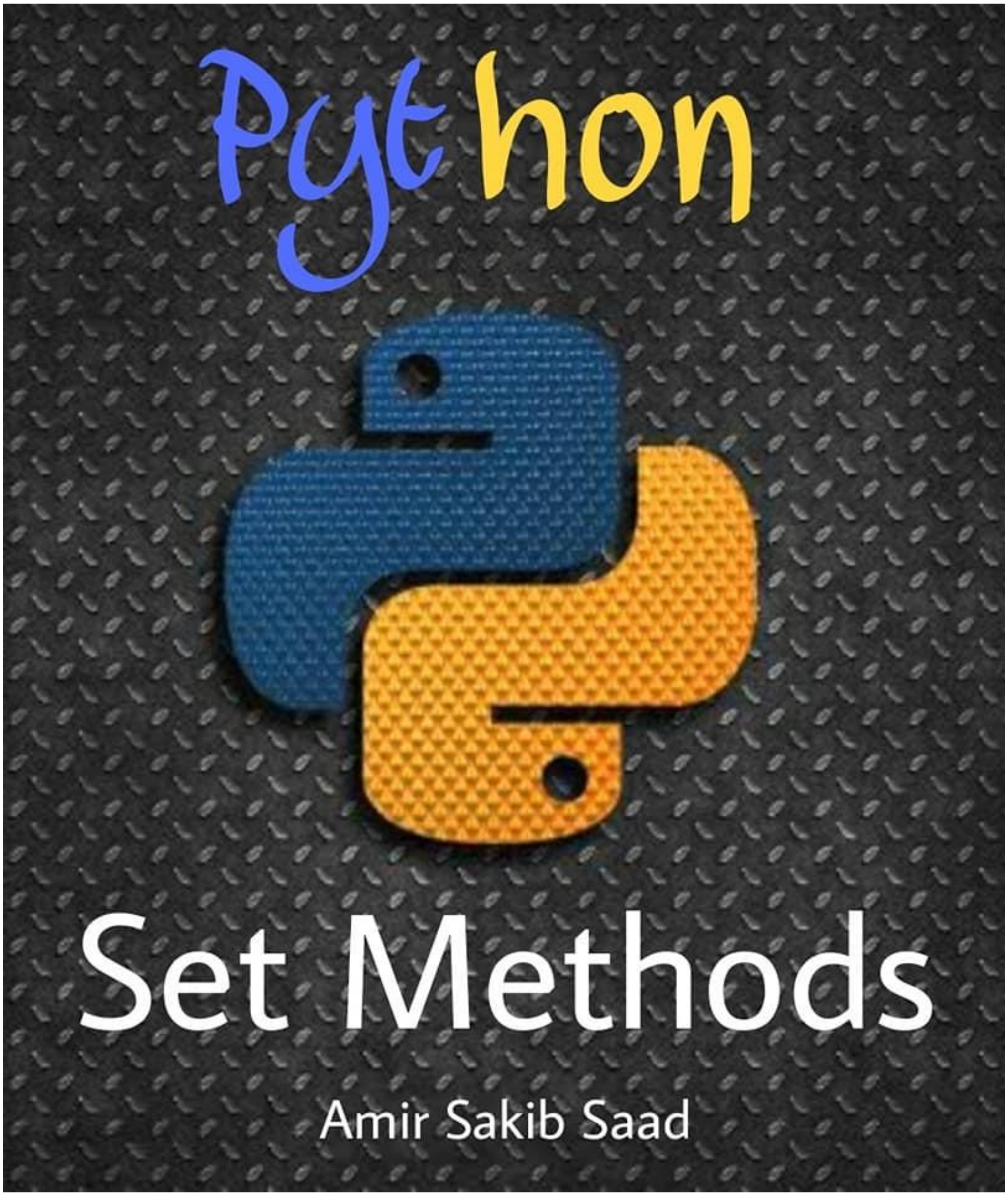
```
# The count() method returns the number of times a specified value appears in the tuple.

tuple = (1,3,5,7,5,8,6,4)

x = tuple.count(5)
print(x)
```

```
2
```

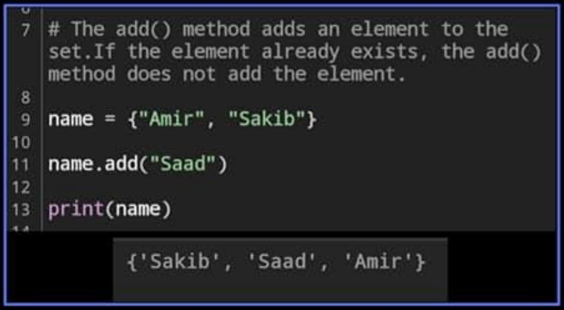
index( )

```
# The index() method finds the first occurrence of the specified value.The index() method raises an exception if the value is not found.

tuple = (1,3,5,7,4,8,6,8,6,8,9)

x = tuple.index(8)
print(x)
```

```
5
```


Python
Set Methods
Amir Sakib Saad

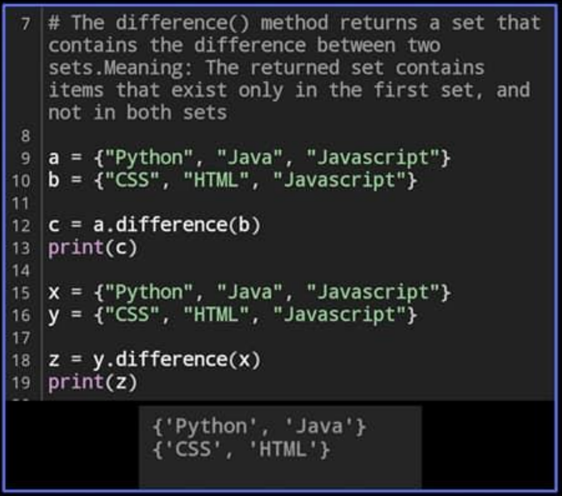
add( )



```
# The add() method adds an element to the set.If the element already exists, the add() method does not add the element.

name = {"Amir", "Sakib"}

name.add("Saad")

print(name)
```

```
{'Sakib', 'Saad', 'Amir'}
```

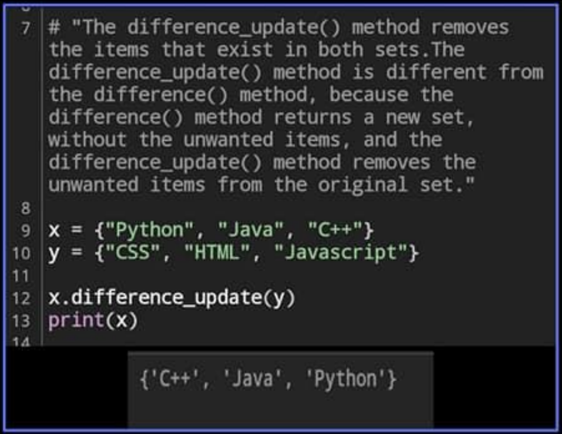
difference( )



```
# 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

a = {"Python", "Java", "Javascript"}
b = {"CSS", "HTML", "Javascript"}

c = a.difference(b)
print(c)

x = {"Python", "Java", "Javascript"}
y = {"CSS", "HTML", "Javascript"}

z = y.difference(x)
print(z)
```

```
{'Python', 'Java'}
{'CSS', 'HTML'}
```

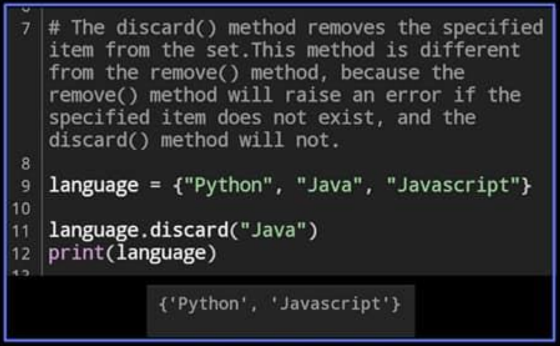
difference_
update( )



```
# "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."

x = {"Python", "Java", "C++"}
y = {"CSS", "HTML", "Javascript"}

x.difference_update(y)
print(x)
```

```
{'C++', 'Java', 'Python'}
```

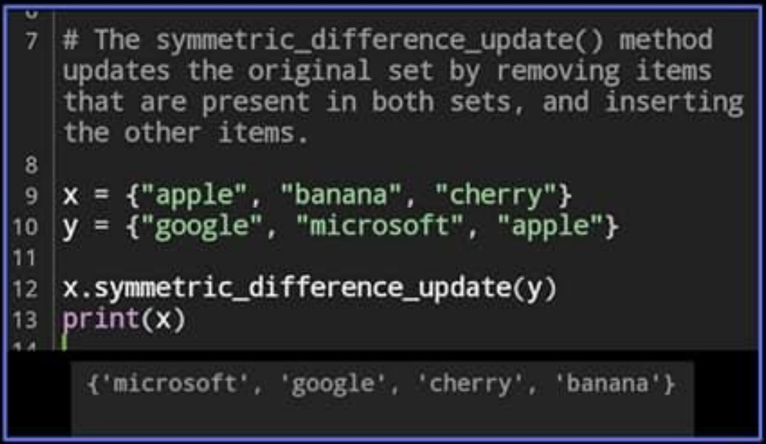
discard( )



```
# 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.

language = {"Python", "Java", "Javascript"}

language.discard("Java")
print(language)
```

```
{'Python', 'Javascript'}
```

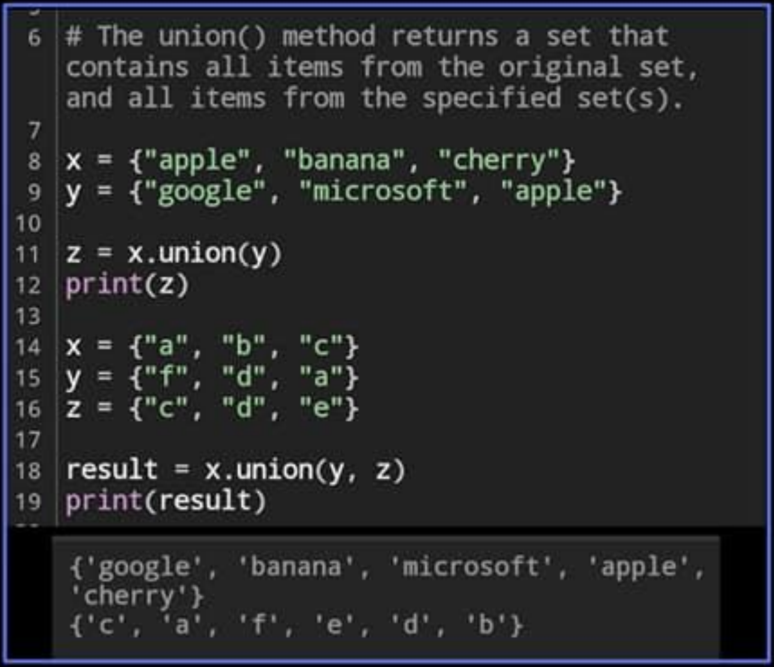
symmetric_
difference_
update( )



```
# The symmetric_difference_update() method updates the original set by removing items that are present in both sets, and inserting the other items.

x = {"apple", "banana", "cherry"}
y = {"google", "microsoft", "apple"}

x.symmetric_difference_update(y)
print(x)
```

```
{'microsoft', 'google', 'cherry', 'banana'}
```

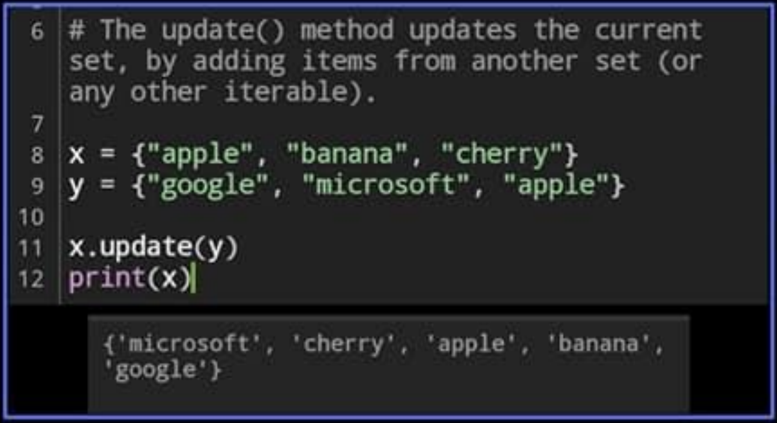
union( )



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

x = {"apple", "banana", "cherry"}
y = {"google", "microsoft", "apple"}

z = x.union(y)
print(z)

x = {"a", "b", "c"}
y = {"f", "d", "a"}
z = {"c", "d", "e"}

result = x.union(y, z)
print(result)
```

```
{'google', 'banana', 'microsoft', 'apple', 'cherry'}
{'c', 'a', 'f', 'e', 'd', 'b'}
```

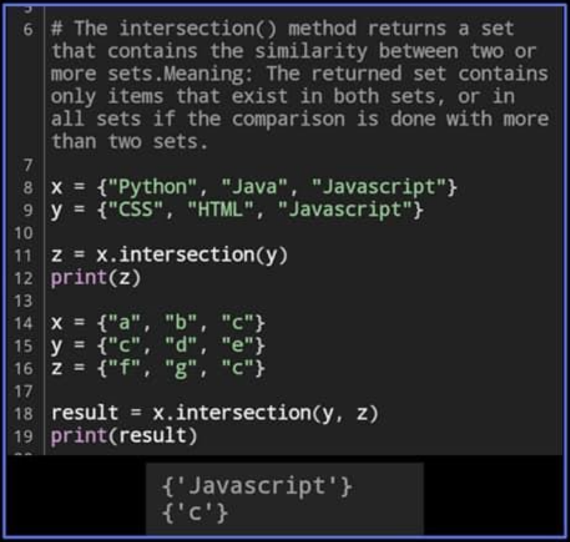
update( )



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

x = {"apple", "banana", "cherry"}
y = {"google", "microsoft", "apple"}

x.update(y)
print(x)
```

```
{'microsoft', 'cherry', 'apple', 'banana', 'google'}
```

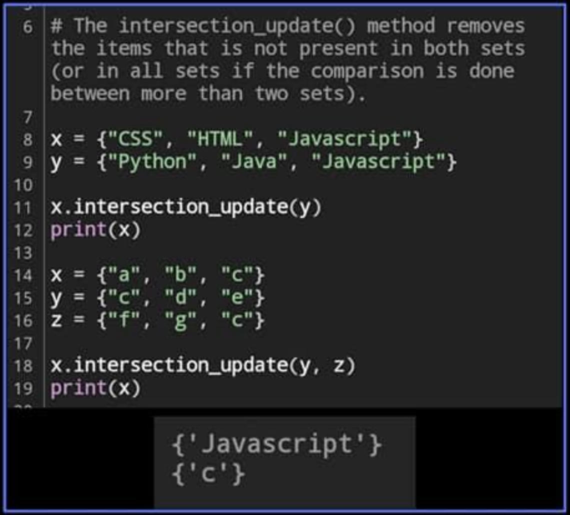
intersection( )



```python
# 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.

x = {"Python", "Java", "Javascript"}
y = {"CSS", "HTML", "Javascript"}

z = x.intersection(y)
print(z)

x = {"a", "b", "c"}
y = {"c", "d", "e"}
z = {"f", "g", "c"}

result = x.intersection(y, z)
print(result)
```

```
{'Javascript'}
{'c'}
```

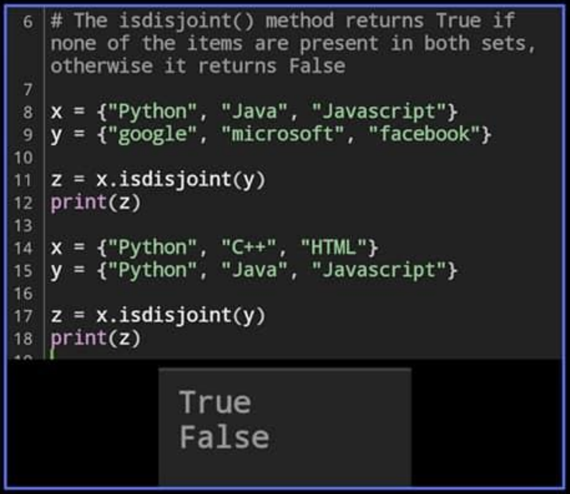
intersection_
update( )



```python
# 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).

x = {"CSS", "HTML", "Javascript"}
y = {"Python", "Java", "Javascript"}

x.intersection_update(y)
print(x)

x = {"a", "b", "c"}
y = {"c", "d", "e"}
z = {"f", "g", "c"}

x.intersection_update(y, z)
print(x)
```

```
{'Javascript'}
{'c'}
```

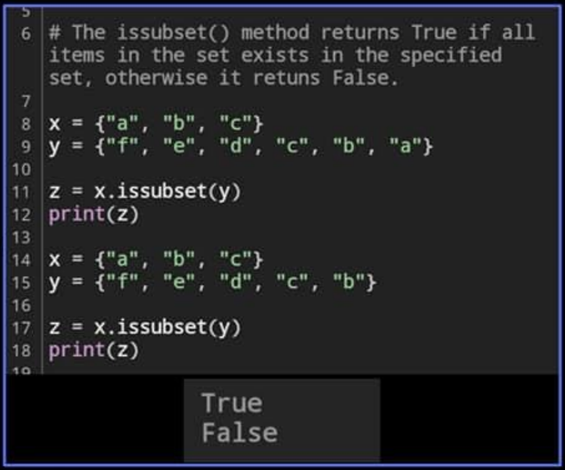
isdisjoint( )



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

x = {"Python", "Java", "Javascript"}
y = {"google", "microsoft", "facebook"}

z = x.isdisjoint(y)
print(z)

x = {"Python", "C++", "HTML"}
y = {"Python", "Java", "Javascript"}

z = x.isdisjoint(y)
print(z)
```

```
True
False
```

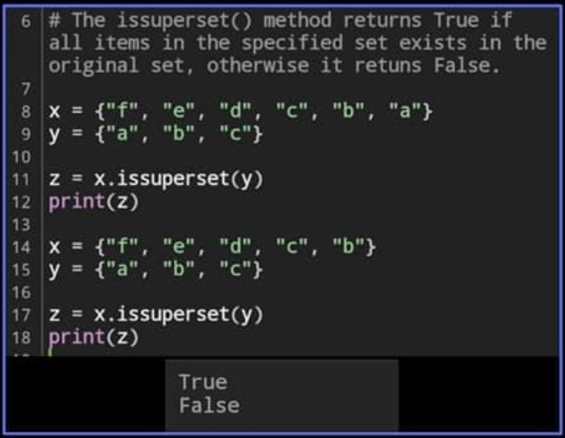
issubset( )



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

x = {"a", "b", "c"}
y = {"f", "e", "d", "c", "b", "a"}

z = x.issubset(y)
print(z)

x = {"a", "b", "c"}
y = {"f", "e", "d", "c", "b"}

z = x.issubset(y)
print(z)
```

```
True
False
```

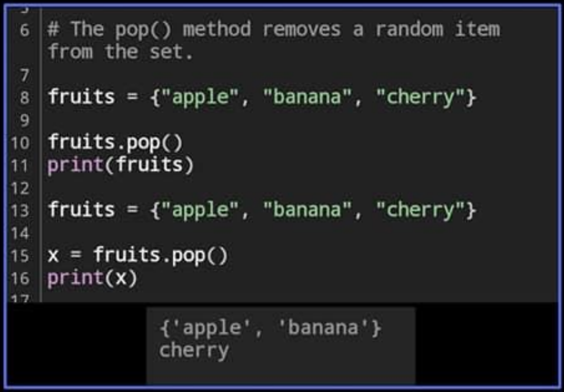
issuperset( )



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

x = {"f", "e", "d", "c", "b", "a"}
y = {"a", "b", "c"}

z = x.issuperset(y)
print(z)

x = {"f", "e", "d", "c", "b"}
y = {"a", "b", "c"}

z = x.issuperset(y)
print(z)
```

```
True
False
```

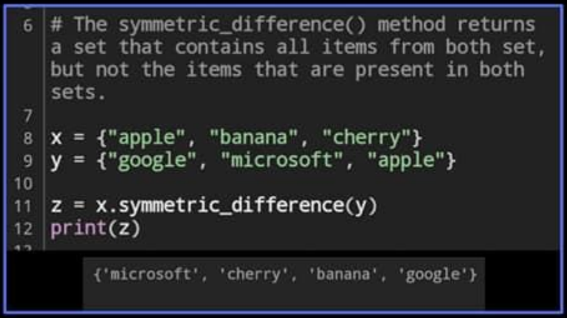
pop( )



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

fruits = {"apple", "banana", "cherry"}

fruits.pop()
print(fruits)

fruits = {"apple", "banana", "cherry"}

x = fruits.pop()
print(x)
```

```
{'apple', 'banana'}
cherry
```

symmetric_difference( )



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

x = {"apple", "banana", "cherry"}
y = {"google", "microsoft", "apple"}

z = x.symmetric_difference(y)
print(z)
```

```
{'microsoft', 'cherry', 'banana', 'google'}
```