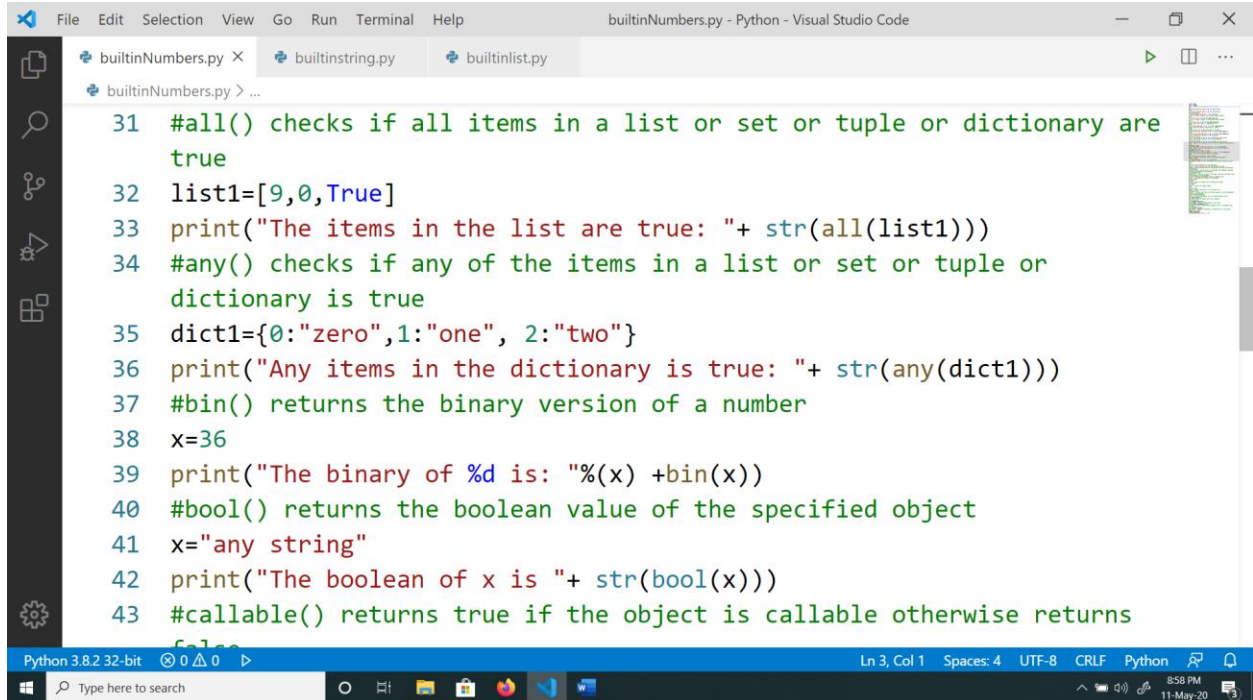
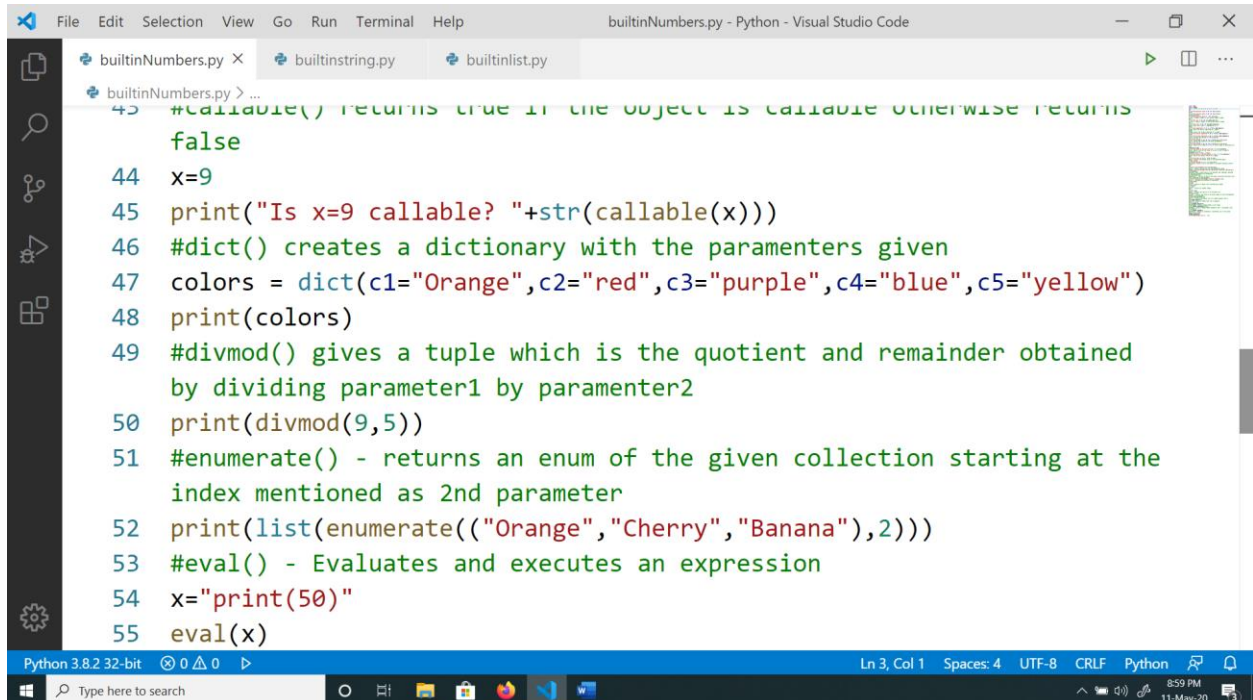


```
1 import math
2 import random
3 #abs() returns the absolute value of a number
4 x=9
5 print("The absolute value of %d: %d" %(x,abs(x)))
6 x=-78
7 print("The absolute value of %d: %d" %(x,abs(x)))
8 x=complex(19,25)
9 print("The absolute value of : %f" %(abs(x)))
10 # ceil() rounds a number to the nearest higher integer
11 x=8.3
12 print("The ceil of %d: %d" %(x,math.ceil(x)))
13 # floor() rounds a number to the nearest lowest integer
14 x=8.3
15 print("The floor of %d: %d" %(x,math.floor(x)))
```

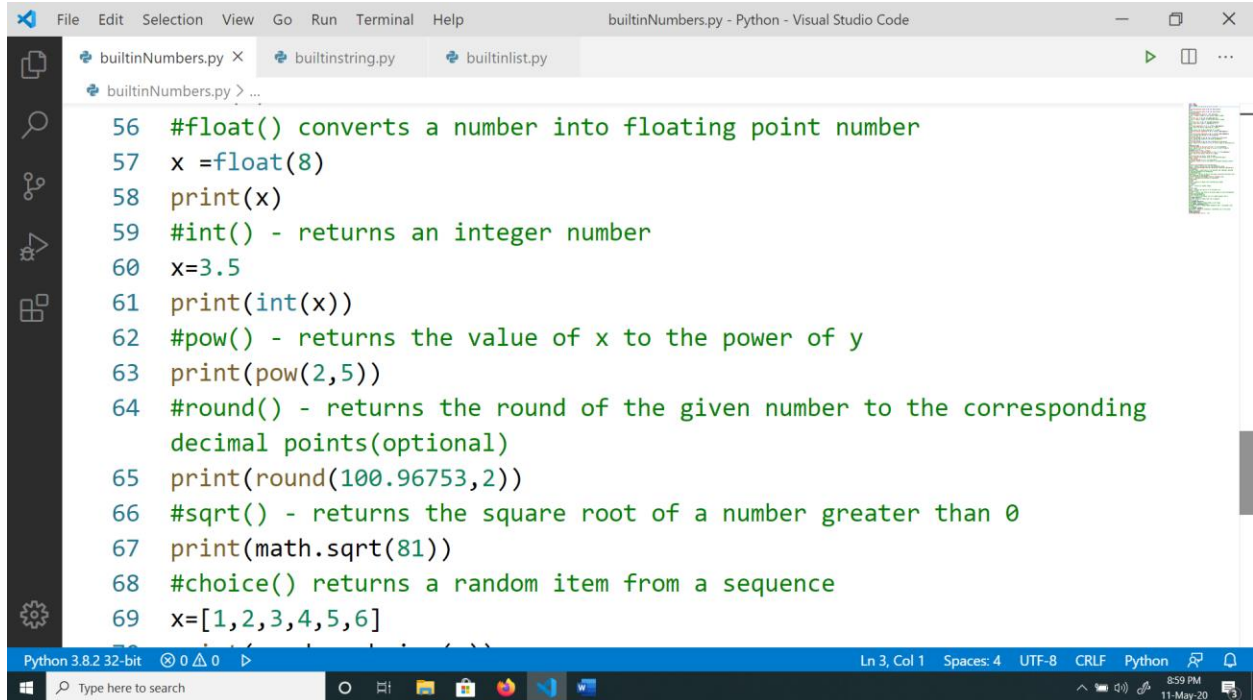
```
16 #exp() gives the e power x (Exponential of x)
17 x=45.7
18 print("The exponential of %f is: %f"%(x, math.exp(x)))
19 #log() gives the natural logarithm of a number
20 x=100
21 #log10() gives the 10 base logarithm of a number
22 print("The natural logarithm of %d is: %f"%(x, math.log(x)))
23 x=100
24 print("The 10 base logarithm of %d is: %f"%(x, math.log10(x)))
25 # max() returns the maximum of the parameters
26 x,y,z = 45, 90, 21
27 print("The maximum of %d, %d, %d is %d"%(x,y,z,max(x,y,z)))
28 #min() returns the minimum of the given parameters
29 x,y,z = 45, 90, 21
30 print("The minimum of %d, %d, %d is %d"%(x,y,z,min(x,y,z)))
```



```
31 #all() checks if all items in a list or set or tuple or dictionary are
    true
32 list1=[9,0,True]
33 print("The items in the list are true: "+ str(all(list1)))
34 #any() checks if any of the items in a list or set or tuple or
    dictionary is true
35 dict1={0:"zero",1:"one", 2:"two"}
36 print("Any items in the dictionary is true: "+ str(any(dict1)))
37 #bin() returns the binary version of a number
38 x=36
39 print("The binary of %d is:"%(x) +bin(x))
40 #bool() returns the boolean value of the specified object
41 x="any string"
42 print("The boolean of x is "+ str(bool(x)))
43 #callable() returns true if the object is callable otherwise returns
```



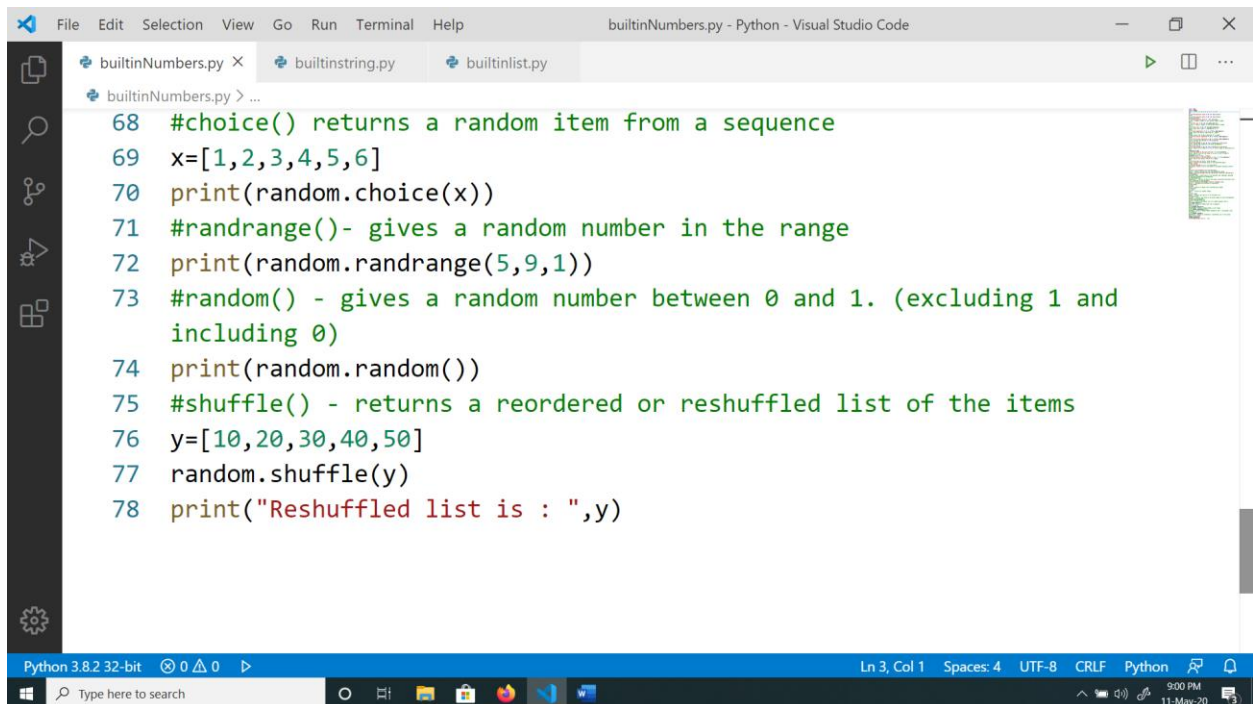
```
43 #callable() returns true if the object is callable otherwise returns
    false
44 x=9
45 print("Is x=9 callable? "+str(callable(x)))
46 #dict() creates a dictionary with the parameters given
47 colors = dict(c1="Orange",c2="red",c3="purple",c4="blue",c5="yellow")
48 print(colors)
49 #divmod() gives a tuple which is the quotient and remainder obtained
    by dividing parameter1 by parameter2
50 print(divmod(9,5))
51 #enumerate() - returns an enum of the given collection starting at the
    index mentioned as 2nd parameter
52 print(list(enumerate(("Orange","Cherry","Banana"),2)))
53 #eval() - Evaluates and executes an expression
54 x="print(50)"
55 eval(x)
```



This screenshot shows a Visual Studio Code window with the file 'builtinNumbers.py' open. The code defines several Python built-in functions with their descriptions and usage examples. The functions include float(), int(), pow(), round(), sqrt(), and choice(). The code is as follows:

```
56 #float() converts a number into floating point number
57 x =float(8)
58 print(x)
59 #int() - returns an integer number
60 x=3.5
61 print(int(x))
62 #pow() - returns the value of x to the power of y
63 print(pow(2,5))
64 #round() - returns the round of the given number to the corresponding
    decimal points(optional)
65 print(round(100.96753,2))
66 #sqrt() - returns the square root of a number greater than 0
67 print(math.sqrt(81))
68 #choice() returns a random item from a sequence
69 x=[1,2,3,4,5,6]
```

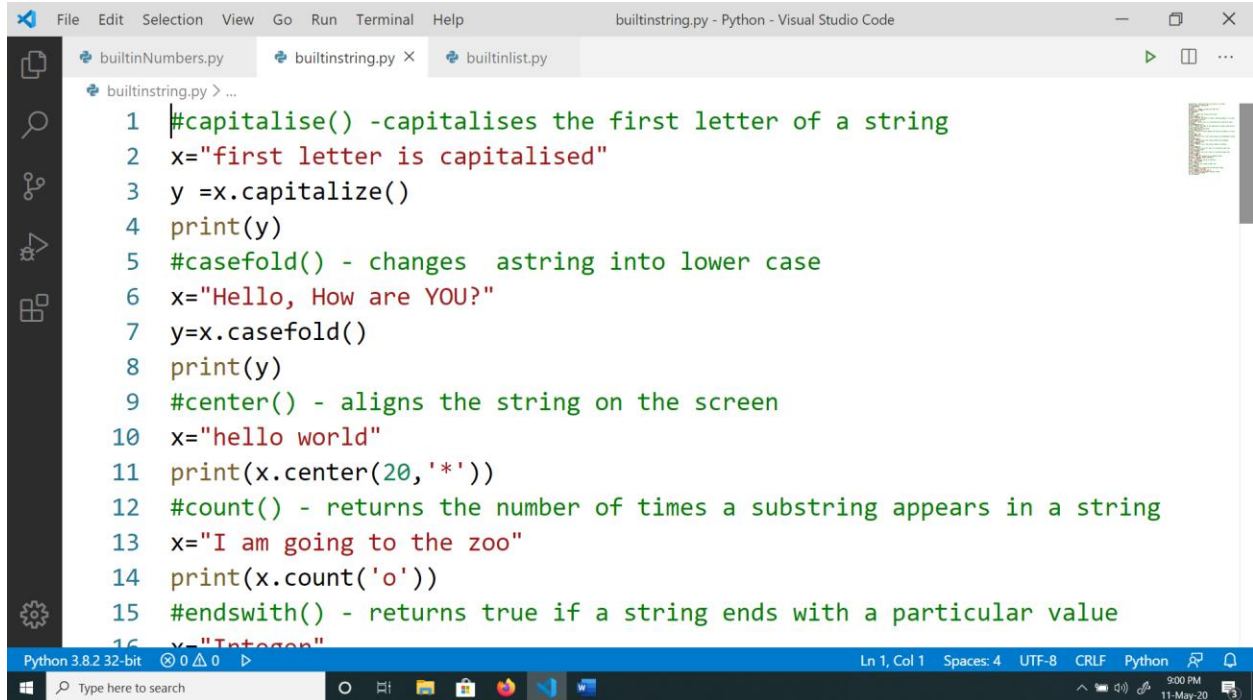
The status bar at the bottom indicates 'Python 3.8.2 32-bit', 'Ln 3, Col 1', 'Spaces: 4', 'UTF-8', 'CRLF', and 'Python'. The system tray shows the time as 8:59 PM on 11-May-20.



This screenshot shows a Visual Studio Code window with the file 'builtinNumbers.py' open. The code continues with functions from the random module: choice(), randrange(), random(), and shuffle(). The code is as follows:

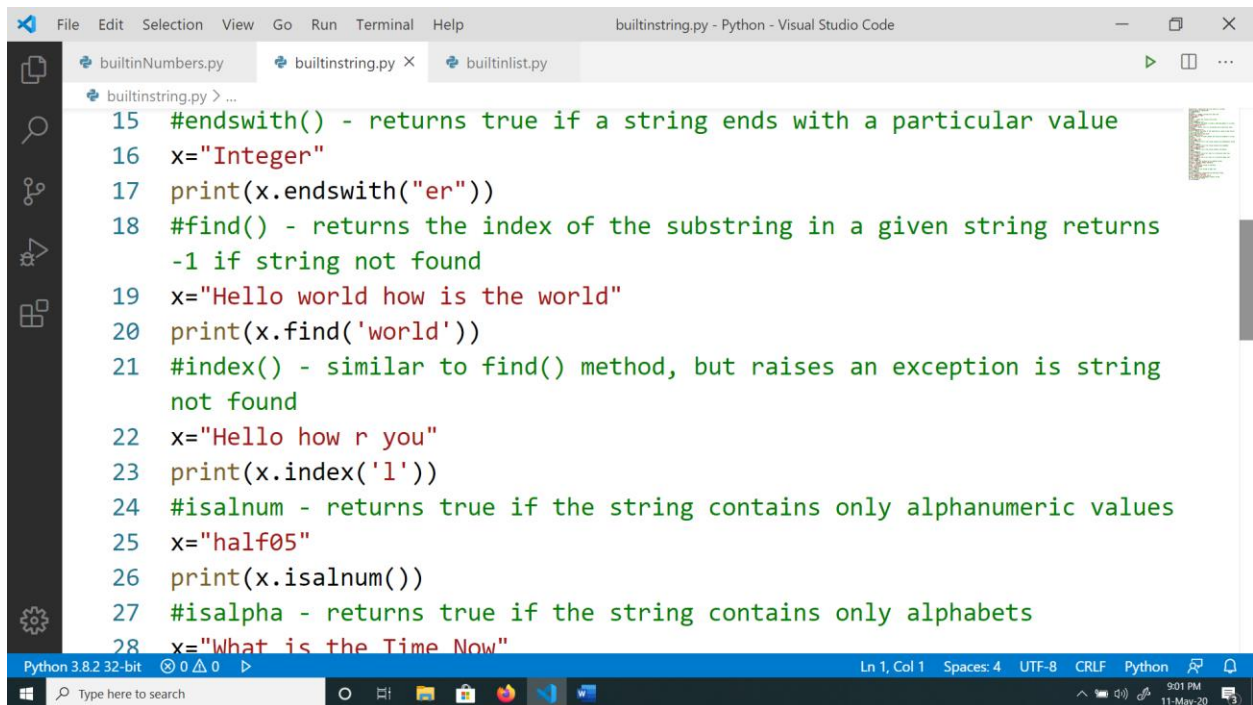
```
68 #choice() returns a random item from a sequence
69 x=[1,2,3,4,5,6]
70 print(random.choice(x))
71 #randrange()- gives a random number in the range
72 print(random.randrange(5,9,1))
73 #random() - gives a random number between 0 and 1. (excluding 1 and
    including 0)
74 print(random.random())
75 #shuffle() - returns a reordered or reshuffled list of the items
76 y=[10,20,30,40,50]
77 random.shuffle(y)
78 print("Reshuffled list is : ",y)
```

The status bar at the bottom indicates 'Python 3.8.2 32-bit', 'Ln 3, Col 1', 'Spaces: 4', 'UTF-8', 'CRLF', and 'Python'. The system tray shows the time as 9:00 PM on 11-May-20.



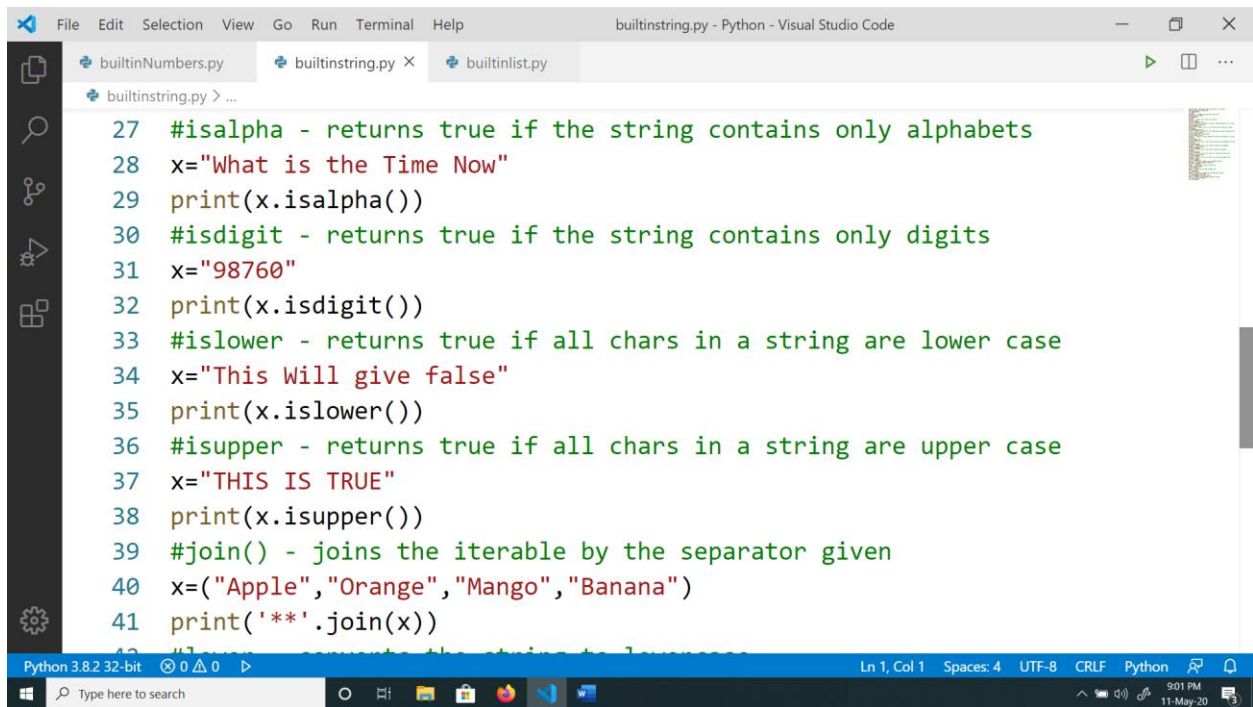
```
1 #capitalize() -capitalises the first letter of a string
2 x="first letter is capitalised"
3 y =x.capitalize()
4 print(y)
5 #casefold() - changes astring into lower case
6 x="Hello, How are YOU?"
7 y=x.casefold()
8 print(y)
9 #center() - aligns the string on the screen
10 x="hello world"
11 print(x.center(20,' '))
12 #count() - returns the number of times a substring appears in a string
13 x="I am going to the zoo"
14 print(x.count('o'))
15 #endswith() - returns true if a string ends with a particular value
16 x="Integer"
```

Python 3.8.2 32-bit Ln 1, Col 1 Spaces: 4 UTF-8 CRLF Python 11-May-20



```
15 #endswith() - returns true if a string ends with a particular value
16 x="Integer"
17 print(x.endswith("er"))
18 #find() - returns the index of the substring in a given string returns
-1 if string not found
19 x="Hello world how is the world"
20 print(x.find('world'))
21 #index() - similar to find() method, but raises an exception is string
not found
22 x="Hello how r you"
23 print(x.index('l'))
24 #isalnum - returns true if the string contains only alphanumeric values
25 x="half05"
26 print(x.isalnum())
27 #isalpha - returns true if the string contains only alphabets
28 x="What is the Time Now"
```

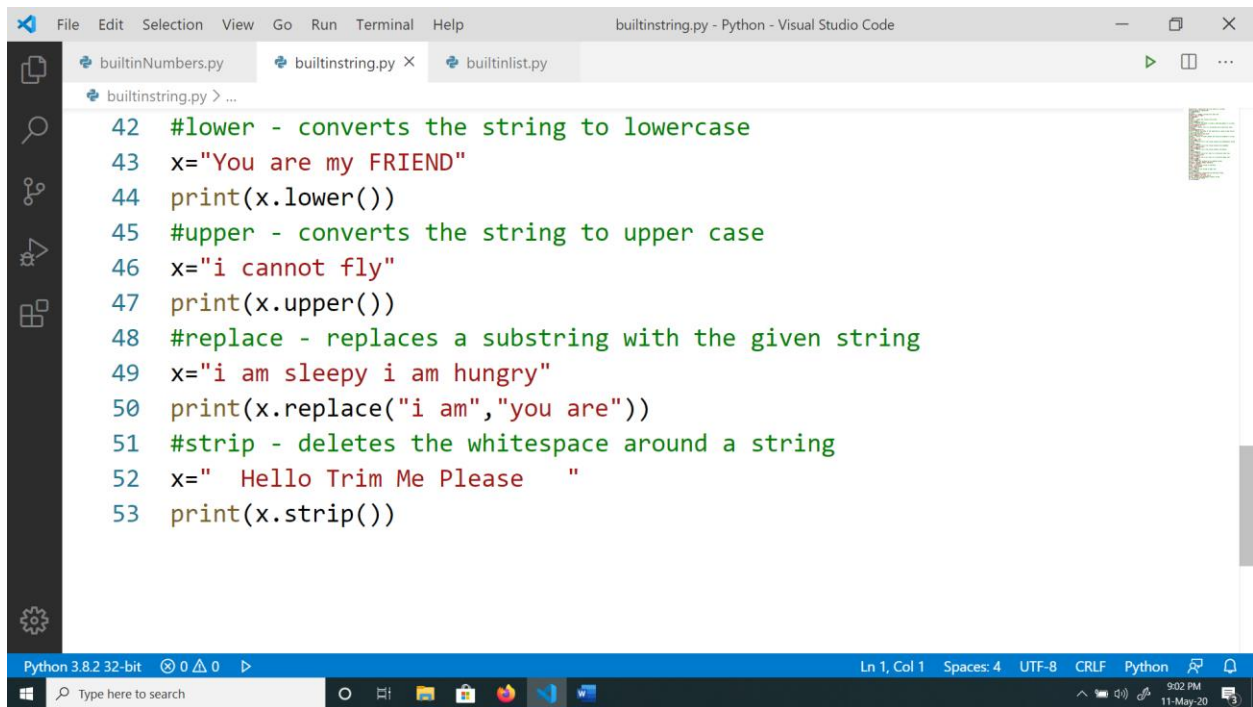
Python 3.8.2 32-bit Ln 1, Col 1 Spaces: 4 UTF-8 CRLF Python 11-May-20



The screenshot shows the Visual Studio Code editor with a Python file named `builtinstring.py`. The code demonstrates several string methods: `isalpha()`, `isdigit()`, `islower()`, `isupper()`, and `join()`. The code is as follows:

```
27 #isalpha - returns true if the string contains only alphabets
28 x="What is the Time Now"
29 print(x.isalpha())
30 #isdigit - returns true if the string contains only digits
31 x="98760"
32 print(x.isdigit())
33 #islower - returns true if all chars in a string are lower case
34 x="This Will give false"
35 print(x.islower())
36 #isupper - returns true if all chars in a string are upper case
37 x="THIS IS TRUE"
38 print(x.isupper())
39 #join() - joins the iterable by the separator given
40 x=("Apple", "Orange", "Mango", "Banana")
41 print('**'.join(x))
```

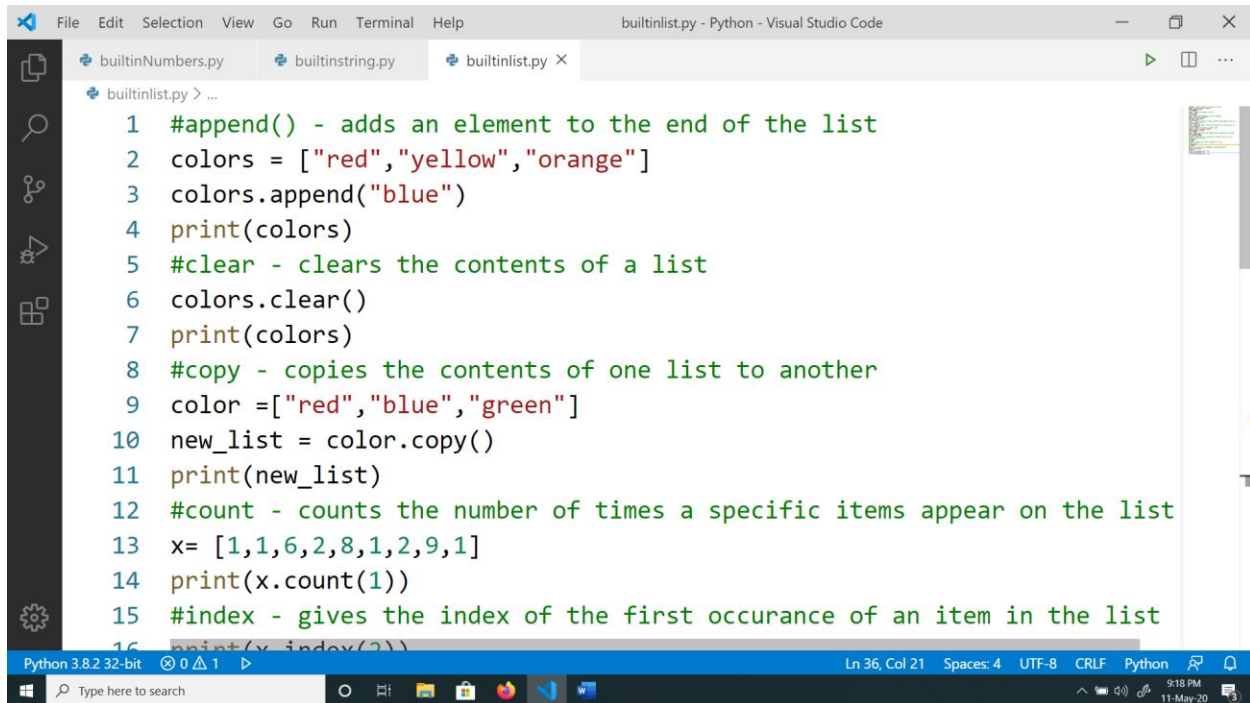
The status bar at the bottom indicates the file is using Python 3.8.2 32-bit, with 4 spaces, UTF-8 encoding, and CRLF line endings. The system clock shows 9:01 PM on 11-May-20.



The screenshot shows the Visual Studio Code editor with the same Python file `builtinstring.py`. The code continues with more string methods: `lower()`, `upper()`, `replace()`, and `strip()`. The code is as follows:

```
42 #lower - converts the string to lowercase
43 x="You are my FRIEND"
44 print(x.lower())
45 #upper - converts the string to upper case
46 x="i cannot fly"
47 print(x.upper())
48 #replace - replaces a substring with the given string
49 x="i am sleepy i am hungry"
50 print(x.replace("i am", "you are"))
51 #strip - deletes the whitespace around a string
52 x=" Hello Trim Me Please "
53 print(x.strip())
```

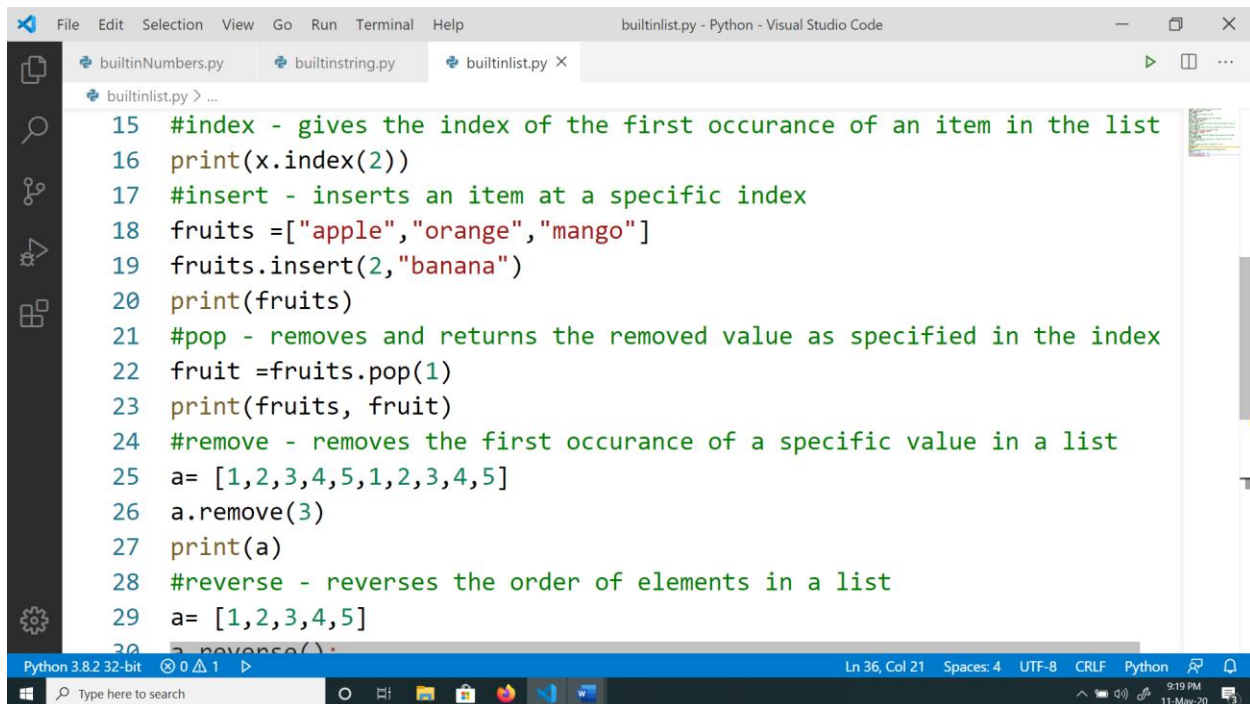
The status bar at the bottom shows the same configuration as the first screenshot, with the system clock now at 9:02 PM on 11-May-20.



This screenshot shows a Visual Studio Code editor window with the file 'builtinlist.py' open. The code demonstrates several list methods: `append()` to add an element, `clear()` to empty a list, `copy()` to create a shallow copy, `count()` to count occurrences, and `index()` to find the index of an element. The code is as follows:

```
1 #append() - adds an element to the end of the list
2 colors = ["red","yellow","orange"]
3 colors.append("blue")
4 print(colors)
5 #clear - clears the contents of a list
6 colors.clear()
7 print(colors)
8 #copy - copies the contents of one list to another
9 color =["red","blue","green"]
10 new_list = color.copy()
11 print(new_list)
12 #count - counts the number of times a specific items appear on the list
13 x= [1,1,6,2,8,1,2,9,1]
14 print(x.count(1))
15 #index - gives the index of the first occurrence of an item in the list
16 print(x.index(2))
```

The status bar at the bottom indicates 'Python 3.8.2 32-bit', 'Ln 36, Col 21', 'Spaces: 4', 'UTF-8', 'CRLF', and 'Python'. The system tray shows the time as 9:18 PM on 11-May-20.



This screenshot shows the same Visual Studio Code editor window with 'builtinlist.py' open, displaying the continuation of the list operations code: `insert()` to add an element at a specific index, `pop()` to remove and return an element, `remove()` to remove a specific value, and `reverse()` to reverse the list order. The code is as follows:

```
15 #index - gives the index of the first occurrence of an item in the list
16 print(x.index(2))
17 #insert - inserts an item at a specific index
18 fruits=["apple","orange","mango"]
19 fruits.insert(2,"banana")
20 print(fruits)
21 #pop - removes and returns the removed value as specified in the index
22 fruit =fruits.pop(1)
23 print(fruits, fruit)
24 #remove - removes the first occurrence of a specific value in a list
25 a= [1,2,3,4,5,1,2,3,4,5]
26 a.remove(3)
27 print(a)
28 #reverse - reverses the order of elements in a list
29 a= [1,2,3,4,5]
30 a.reverse()
```

The status bar at the bottom indicates 'Python 3.8.2 32-bit', 'Ln 36, Col 21', 'Spaces: 4', 'UTF-8', 'CRLF', and 'Python'. The system tray shows the time as 9:19 PM on 11-May-20.

```
28 #reverse - reverses the order of elements in a list
29 a= [1,2,3,4,5]
30 a.reverse();
31 print(a)
32 #sort - sorts the array in ascending or descending order\
33 a=[100,700,300,400]
34 a.sort()
35 print("In ascending order: ",a)
36 a.sort(reverse=True)
37 print("In descending order: ",a)
```

```
1 #count - counts the number of occurrence of an item in a
  tuple
2 x=(9,5,6,3,2,1,8,3,2,3)
3 print(x.count(3))
4 #index - returns the first index of the occurrence of an
  item in a tuple
5 x=("banana","mango","apple","grapes","mango")
6 print(x.index("mango"))
```



```
File Edit Selection View Go Run Terminal Help builtinSets.py - Python - Visual Studio Code

EXPLORER
OPEN EDITORS
  builtinNumbers.py
  builtinstring.py
  builtinlist.py 1
  builtinTuples.py
  builtinSets.py
PYTHON
  > Class1Assignment
  2014_Book_PythonPr...
  2015_Book_DataStruc...
  2019_Book_Advanced...
  builtinlist.py 1
  builtinNumbers.py
  builtinSets.py
  builtinstring.py
  builtinTuples.py
OUTLINE
TIMELINE

builtinSets.py > ...
1 #add() - adds an element to the set if it doesnot already exists in the set
2 x = {"apple", "banana", "cherry"}
3 x.add("orange")
4 print(x)
5 #difference() - returns the elements in first set which are not in the second set into a new set
6 x = {1, 2, 3}
7 y = {3, 4, 5}
8 z = x.difference(y)
9 print(z)
10 #difference_update() - updates the first set by deleting the contents in the second set which are present in first set
11 x = {1, 2, 3}
12 y = {3, 4, 5}

Python 3.8.2 32-bit 0 1 2 Ln 24, Col 9 Spaces: 4 UTF-8 CRLF Python 9:38 PM 11-May-20
```

```
File Edit Selection View Go Run Terminal Help builtinSets.py - Python - Visual Studio Code

EXPLORER
OPEN EDITORS
  builtinNumbers.py
  builtinstring.py
  builtinlist.py 1
  builtinTuples.py
  builtinSets.py
PYTHON
  > Class1Assignment
  2014_Book_PythonPr...
  2015_Book_DataStruc...
  2019_Book_Advanced...
  builtinlist.py 1
  builtinNumbers.py
  builtinSets.py
  builtinstring.py
  builtinTuples.py
OUTLINE
TIMELINE

builtinSets.py > ...
11 x = {1, 2, 3}
12 y = {3, 4, 5}
13 x.difference_update(y)
14 print(x)
15 #union - Gives the union of two sets
16 x={1,2,3}
17 y={3,4,5}
18 z=x.union(y)
19 print(z)
20 #intersection - Gives the intersection of two sets
21 x={1,2,3}
22 y={3,4,5}
23 z=x.intersection(y)
24 print(z)
25 #update - updates the current set by adding items from

Python 3.8.2 32-bit 0 1 2 Ln 24, Col 9 Spaces: 4 UTF-8 CRLF Python 9:38 PM 11-May-20
```



```
File Edit Selection View Go Run Terminal Help builtinSets.py - Python - Visual Studio Code

EXPLORER
OPEN EDITORS
  builtinNumbers.py
  builtinstring.py
  builtinlist.py 1
  builtinTuples.py
  builtinSets.py
PYTHON
  > Class1Assignment
  2014_Book_PythonPr...
  2015_Book_DataStruc...
  2019_Book_Advanced...
  builtinlist.py 1
  builtinNumbers.py
  builtinSets.py
  builtinstring.py
  builtinTuples.py
OUTLINE
TIMELINE

25 #update - updates the current set by adding items from
    second set not present in first set
26 x={1,2,3}
27 y={3,4,5}
28 z=x.update(y)
29 #discard - removes the item from the set
30 fruits= {"apple","banana","mango"}
31 fruits.discard("apple")
32 print(fruits)

Python 3.8.2 32-bit 0 1 Ln 24, Col 9 Spaces: 4 UTF-8 CRLF Python 9:38 PM 11-May-20
```

```
File Edit Selection View Go Run Terminal Help builtinDictionary.py - Python - Visual Studio Code

EXPLORER
OPEN EDITORS
  builtinNumbers.py
  builtinstring.py
  builtinlist.py 1
  builtinTuples.py
  builtinSets.py
  builtinDictionary.py
PYTHON
  > Class1Assignment
  2014_Book_PythonPr...
  2015_Book_DataStruc...
  2019_Book_Advanced...
  builtinlist.py 1
  builtinNumbers.py
  builtinSets.py
  builtinstring.py
  builtinTuples.py
OUTLINE
TIMELINE

1 #items - returns the items in a dictionary as a view
  object
2 car = {
3     "brand": "Fiat",
4     "model": "Premier Padmini",
5     "year": 1960
6 }
7 x = car.items()
8 print(x)
9 #keys - returns the keys of the dictionary as a view
  object
10 car = {
11     "brand": "Fiat",
12     "model": "Premier Padmini",
13     "year": 1960
14 }

Python 3.8.2 32-bit 0 1 Ln 39, Col 31 Spaces: 4 UTF-8 CRLF Python 9:50 PM 11-May-20
```

```
14 }
15 x = car.keys()
16 print(x)
17 #values() - Returns the values of a dictionary as a view
   object
18 car = {
19     "brand": "Fiat",
20     "model": "Premier Padmini",
21     "year": 1960
22 }
23 x = car.values()
24 print(x)
25 #get() - Returns the value of the item specified by the
   key
26 car = {
```

```
26 car = {
27     "brand": "Fiat",
28     "model": "Premier Padmini",
29     "year": 1960
30 }
31 x = car.get("year")
32 print(x)
33 #update() - inserts a specified value into the dictionary
34 car = {
35     "brand": "Fiat",
36     "model": "Premier Padmini",
37     "year": 1960
38 }
39 car.update({"color": "Maroon"})
40 print(car)
```