the lines run in Visual Studio:

```
>>> py
>>> 2+2
>>> 7 * 10
>>> divmod(10,4)
>>> width = 60
>>> height = 3 * 7
>>> width * height
>>>  tax = 12.5 / 100
>>> tax
>>> round(_,1)
>>> s = 'First line.\nSecond line.'
>>> s                                   'First line.\nSecond line. '
>>> print(s)                            First line.
                                        Second line.
>>> print('C:\some\name')               C:\some
                                        ame
>>> print(r'C:\some\name')              C:\some\name
>>> 3 * 'un' + 'ium'                   'ununununium'
>>> 'Did' 'Coding'                     'DidCoding'
>>>  word = 'Didcoding'
>>> word[0]  # character in position 0
>>> word[5]  # character in position 5
>>> word[0:2]  # characters from position 0 (included) to 2 (excluded)
>>> word[:2]   # character from the beginning to position 2 (excluded)
>>> word[4:]   # characters from position 4 (included) to the end
>>> word[-2:]  # characters from the second-last (included) to the end
>>> word[:2] + word[2:]
>>> 'P' + word[1:]
>>> s = 'bobby-didcoding'
>>> len(s)
>>> x=20
>>> y=400
>>> repr((x, y, ('spam', 'eggs')))          "(20, 400, ('spam', 'eggs'))"
>>> str((x, y, ('spam', 'eggs')))
>>> print('{0} and {1}'.format('spam', 'eggs'))     spam and eggs
>>> squares = [1, 4, 9, 16, 25]
>>> squares                                    [1, 4, 9, 16, 25]
>>>squares[-3:]                               [9, 16, 25]
>>>squares[:]
>>>squares + [36, 49, 64, 81, 100]
>>> squares.append(216)
```

```
>>> letters = ['a', 'b', 'c', 'd']
>>> len(letters)
```

```
>>> a = ['a', 'b', 'c']
>>> n = [1, 2, 3]
>>> x = [a, n]
>>> x[0]                    ['a', 'b', 'c']
>>> x                       [['a', 'b', 'c'], [1, 2, 3]]
```

```
>>> y = [x**2 for x in range(10)]
>>> y                        [0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

```
>>> x = list(('bobby', 'at', 'didcoding','dot', 'com')) # creates a list object
>>> x                       ['bobby', 'at', 'didcoding', 'dot', 'com']
```

```
>>> t = 12345, 54321, 'hello!'
>>> t                        (12345, 54321, 'hello!')
>>> t[0]                     12345
```

```
>>> v = ([1, 2, 3], [3, 2, 1])
>>> v                        ([1, 2, 3], [3, 2, 1])
```

```
>>> u = t, (1, 2, 3, 4, 5)
>>> u                        ((12345, 54321, 'hello!'), (1, 2, 3, 4, 5))
```

```
>>> x = tuple(['bobby', 'at', 'didcoding','dot', 'com'])
>>> x                        ('bobby', 'at', 'didcoding', 'dot', 'com')
```

```
>>> tuple([x**2 for x in range(10)])          (0, 1, 4, 9, 16, 25, 36, 49, 64, 81)
```

```
>>> basket = {'apple', 'orange', 'apple', 'pear', 'orange', 'banana'}
>>> print(basket)              {'apple', 'banana', 'pear', 'orange'}
>>> 'orange' in basket
```

```
>>> a = set('abracadabra')
>>> b = set('alacazam')
>>> a                        {'b', 'c', 'r', 'd', 'a'}
>>> a – b                    {'b', 'r', 'd'}
>>> a | b
>>> a & b
>>> a ^ b
```

```
>>> some_dict = {
...     'a_key': 'a_value',
...     'a_key_2': 'a_value_2',
...     'a_key_3': ['a_list', 'as', 'a value'],
...     'a_key_4': {'a_dict': 'as a value'}
... }
>>> some_dict       {'a_key': 'a_value', 'a_key_2': 'a_value_2', 'a_key_3': ['a_list', 'as', 'a
value'], 'a_key_4': {'a_dict': 'as a value'}}
>>> some_dict['a_key'] = 'new_value'
```

```
>>> {x: x**2 for x in (2, 4, 6)}              {2: 4, 4: 16, 6: 36}
```

```
>>> x = dict(a=1, b=2, c=3, d=4)
>>> x                                   {'a': 1, 'b': 2, 'c': 3, 'd': 4}
```

```
>>> def number_play(x):
...     if x < 0:
...         x = 0
...         print('Negative changed to zero')
...     elif x == 0:
...         print('Zero')
...     elif x == 1:
...         print('Single')
...     else:
...         print('More')
...
>>> number_play(-1)
Negative changed to zero
>>> number_play(0)
Zero
>>> number_play(1)
Single
>>> number_play(2)
More
```

```
>>> def http_error(status):
...     match status:
...         case 400:
...             return "Bad request"
...         case 404:
...             return "Not found"
...         case 418:
...             return "I'm a teapot"
...         case _:
...             return "Something's wrong with the internet"
...
>>> http_error(418)
"I'm a teapot"
```

```
>>> words = ['cat', 'window', 'defenestrate']
>>> for w in words:
...     print(w, len(w))
...
cat 3
window 6
defenestrate 12
```

```
>>> for n in range(2, 10): #equivalent of...for n in [2,3,4,5,6,7,8,9]:
...     for x in range(2, n): #first loop is for x in range(2, 2):
...         if n % x == 0:
...             print(n, 'equals', x, '*', n//x)
...             break
...     #the else runs when no break clause occurs
...     else:
...         print(n, 'is a prime number')
2 is a prime number
3 is a prime number
4 equals 2 * 2
5 is a prime number
6 equals 2 * 3
7 is a prime number
8 equals 2 * 4
9 equals 3 * 3
```

```
>>> for num in range(2, 10): #equivalent of...for n in [2,3,4,5,6,7,8,9]:
...     if num % 2 == 0:
...         print("Found an even number", num)
...         continue #Will continue with the next loop
...     print("Found an odd number", num)
...
Found an even number 2
Found an odd number 3
Found an even number 4
Found an odd number 5
Found an even number 6
Found an odd number 7
Found an even number 8
Found an odd number 9
```

```
>>> while True:
...     try:
...         x = int(input("Please enter a number: "))
...         break
...     #Add multiple exceptions in a tuple (RuntimeError, TypeError, NameError)
...     except (RuntimeError, TypeError, NameError, ValueError):
...         print("Oops!  That was no valid number.  Try again...")
...
Please enter a number:
Oops!  That was no valid number.  Try again...
Please enter a number:
Oops!  That was no valid number.  Try again...
Please enter a number: 21
```

```
>>> def divide(x, y):
...     try:
...         result = x / y
...     except ZeroDivisionError:
...         print("division by zero!")
...     except TypeError:
...         print("Must be int!")
...     else:
...         print("result is", result)
...     finally:
...         print("executing finally clause")
...
>>> divide(2, 1)
result is 2.0
executing finally clause
```

```
>>> class MyClass:
...     """A simple example class"""
...     i = 12345
...
...     def f(self):
...         return 'hello world'
...
>>> MyClass.i # return the int
12345
>>> MyClass.f # returns a function object
<function MyClass.f at 0x0000022DFA2087C0>
>>> MyClass.__doc__ # magic method/dunder method that return the text literal
'A simple example class'
```