

<p>Summing and multiplying</p> <pre>nums = [1, 2, 3] print(nums + [4, 5, 6]) print(nums * 3)</pre> <p>Lists can be added and multiplied in the same way as strings.</p>	<p>ALL & ANY</p> <pre>nums = [55, 44, 33, 22, 11] if all([i > 5 for i in nums]): print("All larger than 5") if any([i % 2 == 0 for i in nums]): print("At least one is even")</pre> <p>Often used in conditional statements, all and any take a list as an argument, and return True if all or any (respectively) of their arguments evaluate to True (and False otherwise).</p>	<p>ENUMERATE</p> <pre>nums = [55, 44, 33, 22, 11] for v in enumerate(nums): print(v) ----- (0, 55) (1, 44) (2, 33) (3, 22) (4, 11)</pre>
<p>"insert" FUNCTION</p> <pre>words = ["Python", "fun"] index = 1 words.insert(index, "is") print(words) ----- >>> ['Python', 'is', 'fun'] >>></pre>	<p>IN and NOT operator</p> <pre>words = ["spam", "egg", "spam", "sausage"] print("spam" in words) #RETURNS TRUE ----- nums = [1, 2, 3] print(not 4 in nums) #RETURNS TRUE print(4 not in nums)</pre> <p>The in operator is also used to determine whether or not a string is a substring of another string.</p>	<p>The function enumerate iterates through the values of a list and returns the index and value of each element.</p>
<p>insert method is similar to append, except that it allows you to insert a new item at any position in the list, as opposed to just at the end.</p>	<p>"range" FUNCTION</p> <pre>numbers = list(range(5, 20, 2)) print(numbers) ----- >>> [5, 7, 9, 11, 13, 15, 17, 19] >>></pre> <p>* The range function creates a sequential list of numbers.</p> <p>*If range is called with one argument, it produces an object with values from 0 to that argument. If it is called with two arguments, it produces values from the first to the second.</p> <p>*range can have a third argument, which determines the interval of the sequence produced</p>	<p>List comprehension</p> <pre>cubes = [i**3 for i in range(1, 10)] print(cubes) >>> [0, 1, 8, 27, 64, 125, 216, 343, 512, 729] >>></pre> <p>A list comprehension is a compact way to create a list. It consists of an expression followed by a for statement on values in the range function.</p> <pre>evens=[i2 for i in range(1, 10) if i%2== 0] print(evens) >>> [0, 4, 16, 36, 64, 100, 144, 196, 256, 324]</pre>
<p>"index" FUNCTION</p> <pre>letters = ['p', 'q', 'r', 's', 'p', 'u'] print(letters.index('r')) print(letters.index('z')) ----- >>> 2 ValueError: 'z' is not in list >>></pre> <p>index method finds the first occurrence of a list item and returns its index.</p>	<p>Trying to create a list comprehension with an if statement on values in the range function will result in a MemoryError.</p>	



"append" FUNCTION

```
nums = [1, 2, 3]
nums.append(4)
print(nums)
-----
>>>
[1, 2, 3, 4]
>>>
```

This adds an item to the end of an existing list.

"Len" FUNCTION

```
nums = [1, 3, 5, 2, 4]
print(len(nums))
-----
-----
>>>
5
>>>
```

List slicing 1

```
squares = [0, 1, 4, 9, 16,
25, 36, 49, 64, 81]
print(squares[2:6])
print(squares[3:8])
-----
[4, 9, 16, 25]
[9, 16, 25, 36, 49]
```

Basic list slicing involves indexing a list with two colon-separated integers.

List slicing 2

```
squares = [0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
print(squares[::2])
print(squares[2:8:3])
-----
>>>
[0, 4, 16, 36, 64]
[4, 25]
```

>>>

Negative values can be used in list slicing (and normal list indexing). When negative values are used for the first and second values in a slice (or a normal index), they count from the end of the list.

```
squares = [0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

```
print(squares[1:-1])
```

>>>

```
[1, 4, 9, 16, 25, 36, 49, 64]
```

>>>

If a negative value is used for the step, the slice is done backwards.

Using `[::-1]` as a slice is a common and idiomatic way to reverse a list.

List slices can also have a third number, representing the step, to include only alternate values in the slice.

