# PythonBasic_4

## 1. What exactly is [ ]?

```
[] is a emplty list, like a =[]
```

## 2. In a list of values stored in a variable called spam, how would you assign the value hello' as the third value? (Assume [2, 4, 6, 8, 10] are in spam.)

In [1]:
```
# solution by changing the value in index 3
spam = [2, 4, 6, 8, 10]
spam[2] = 'hello'
spam
```

Out[1]:
```
[2, 4, 'hello', 8, 10]
```

In [3]:
```
# solution by inserting value in 3rd index
spam = [2, 4, 6, 8, 10]
spam.insert(2,'hello')
spam
```

Out[3]:
```
[2, 4, 'hello', 6, 8, 10]
```

## Lets pretend the spam includes the list ['a', 'b','c','d'] for the next three queries.

## 3. What is the value of spam[int(int(3 * 2) / 11)]?

In [5]:
```
spam = ['a', 'b','c','d']
spam[int(int('3' * 2) / 11)] # spam[int(33/11)] = spam[3]
```

Out[5]:
```
'd'
```

## 4. What is the value of spam[-1]?

In [32]:
```
spam = ['a', 'b','c','d']
spam[-1] # negative index # d
```

Out[32]:
```
'd'
```

## 5. What is the value of spam[:2]?

In [34]:
```
spam[:2] # c
```

Out[34]:
```
['a', 'b']
```

## Let's pretend bacon has the list [3.14, 'cat' 11, 'cat' True] for the next three questions.

## 6. What is the value of bacon.index('cat')?

```
In [17]: bacon = [3.14, 'cat', 11, 'cat', True]
         bacon.index('cat') # it returns the index of first occurrence of 'cat'

Out[17]: 1
```

## 7. How does bacon.append(99) change the look of the list value in bacon?

```
In [21]: bacon = [3.14, 'cat', 11, 'cat', True]
         bacon.append(99) # append adds the item at the end of the list
         bacon

Out[21]: [3.14, 'cat', 11, 'cat', True, 99]
```

## 8. How does bacon.remove('cat') change the look of the list in bacon?

```
In [22]: bacon = [3.14, 'cat', 11, 'cat', True]
         bacon.remove('cat') # remove first occurrence of item
         bacon

Out[22]: [3.14, 11, 'cat', True]
```

## 9. What are the list concatenation and list replication operators?

```
* -> is list replication operatotor
+ -> is list cancatination operator
```

```
In [24]: l1 = [1,3]
         l2 = [7,9]
         # list concatination
         l1+l2

Out[24]: [1, 3, 7, 9]
```

```
In [25]: l1 = [1,3]

         # list replication
         l1*3

Out[25]: [1, 3, 1, 3, 1, 3]
```

## 10. What is difference between the list methods append() and insert()?

```
append() ->Appends object to the end of the list
insert() -> Insert object before index
```

```
In [26]: bacon = [3.14, 'cat', 11, 'cat', True]
         bacon.append(99) # append adds the item at the end of the list
         bacon
```

Out[26]:
```
[3.14, 'cat', 11, 'cat', True, 99]
```

In [27]:
```
# solution by inserting value in 3rd index
spam = [2, 4, 6, 8, 10]
spam.insert(2,'hello')
spam
```

Out[27]:
```
[2, 4, 'hello', 6, 8, 10]
```

## 11. What are the two methods for removing items from a list?

```
remove(item) - removeds first occurence of a item
pop() - Remove and returns item at index (default last).
del(list) - deletes the list
```

In [28]:
```
bacon = [3.14, 'cat', 11, 'cat', True]
bacon.remove('cat')
bacon
```

Out[28]:
```
[3.14, 11, 'cat', True]
```

In [36]:
```
bacon = [3.14, 'cat', 11, 'cat', True]
bacon.pop()
bacon
```

Out[36]:
```
[3.14, 'cat', 11, 'cat']
```

## 12. Describe how list values and string values are identical.

```
1. Both lists and strings can be passed to len()
2. Have indexes and slices
3. Can be used in for loops
4. Can be concatenated or replicated
5. Can be used with the in and not in operators
```

## 13. What's the difference between tuples and lists?

```
Lists :
are mutable - they can have values added, removed, or changed.
lists use the square brackets, [ and ]
Tuples :
are immutable; they cannot be changed at all.
Tuples are written using parentheses, ( and ) while
```

## 14. How do you type a tuple value that only contains the integer 42?

In [38]:
```
tup = (42,)
tup
```

Out[38]:
```
(42,)
```

## 15. How do you get a list value's tuple form? How do you get a tuple value's list form?

```
By using tuple() and list() functions
```

In [39]:
```
l1 = [2,3]
l = tuple(l1)
l
```

Out[39]:
```
(2, 3)
```

In [40]:
```
t1 = (3,4)
t = list(t1)
t
```

Out[40]:
```
[3, 4]
```

## 16. Variables that "contain" list values are not necessarily lists themselves. Instead, what do they contain?

```
They contain references to list values
```

## 17. How do you distinguish between copy.copy() and copy.deepcopy()?

```
The copy.copy() function will do a shallow copy of a list,
The copy.deepcopy() function will do a deep copy of a list. only
copy.deepcopy() will duplicate any lists inside the
list
```