PYTHON

Mohan M J

List Methods

```
>>> fruits = ['orange', 'apple', 'pear', 'banana', 'kiwi', 'apple', 'banana']
>>> fruits.count('apple')                                                  2
>>> fruits.count('tangerine')                                              0
>>> fruits.index('banana')                                                 3
>>> fruits.index('banana', 4)  # Find next banana starting a position 4     6
>>> fruits.reverse()
>>> fruits    ['banana', 'apple', 'kiwi', 'banana', 'pear', 'apple', 'orange']
>>> fruits.append('grape')
>>> fruits    ['banana', 'apple', 'kiwi', 'banana', 'pear', 'apple', 'orange', 'grape']
>>> fruits.sort()
>>> fruits    ['apple', 'apple', 'banana', 'banana', 'grape', 'kiwi', 'orange', 'pear']
>>> fruits.pop()                           'pear'
```

List Comprehensions

```
#List comprehensions provide a concise way to create lists
>>> squares = []
>>> for x in range(10):
...     squares.append(x**2)
...
>>> squares                         [0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
>>> squares = list(map(lambda x: x**2, range(10)))
#or, equivalently:
>>> squares = [x**2 for x in range(10)]
```

List Comprehensions

```
>>> [(x, y) for x in [1,2,3] for y in [3,1,4] if x != y]
                             [(1, 3), (1, 4), (2, 3), (2, 1), (2, 4), (3, 1), (3, 4)]

#it’s equivalent to:
>>> combs = []
>>> for x in [1,2,3]:
...     for y in [3,1,4]:
...         if x != y:
...             combs.append((x, y))
...
>>> combs                    [(1, 3), (1, 4), (2, 3), (2, 1), (2, 4), (3, 1), (3, 4)]
```

List Comprehensions

```
>>> vec = [-4, -2, 0, 2, 4]
>>> [x*2 for x in vec] # create a new list with the values doubled          [-8, -4, 0, 4, 8]
>>> [x for x in vec if x >= 0] # filter the list to exclude negative numbers [0, 2, 4]
>>> [abs(x) for x in vec] # apply a function to all the elements            [4, 2, 0, 2, 4]
>>> freshfruit = ['  banana', '  loganberry ', 'passion fruit  ']
>>> [weapon.strip() for weapon in freshfruit] # call a method on each element
                                                           ['banana', 'loganberry', 'passion fruit']
>>> [(x, x**2) for x in range(6)] # create a list of 2-tuples like (number, square)
                                                   [(0, 0), (1, 1), (2, 4), (3, 9), (4, 16), (5, 25)]
>>> [x, x**2 for x in range(6)] # the tuple must be parenthesized, otherwise an error is raised
>>> vec = [[1,2,3], [4,5,6], [7,8,9]]
>>> [num for elem in vec for num in elem] # flatten a list using a listcomp with two 'for'
                                                                       [1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Looping Techniques

```
#the key and corresponding value can be retrieved at the same time
>>> knights = {'gallahad': 'the pure', 'robin': 'the brave'}
>>> for k, v in knights.items():
...     print(k, v)
#position index and corresponding value can be retrieved at the same time
>>> for i, v in enumerate(['tic', 'tac', 'toe']):
...     print(i, v)
#To loop over two or more sequences at the same time, the entries can be paired
>>> questions = ['name', 'quest', 'favorite color']
>>> answers = ['lancelot', 'the holy grail', 'blue']
>>> for q, a in zip(questions, answers):
...     print('What is your {0}?  It is {1}.'.format(q, a))
```

Looping Techniques

#To loop over a sequence in reverse, first specify the sequence in a forward direction and then call the reversed() function.

```
>>> for i in reversed(range(1, 10, 2)):
...     print(i)
```

#To loop over a sequence in sorted order, use the sorted() function which returns a new sorted list while leaving the source unaltered

```
>>> basket = ['apple', 'orange', 'apple', 'pear', 'orange', 'banana']
>>> for f in sorted(set(basket)):
...     print(f)
```

Exceptions

Even if a statement or expression is syntactically correct, it may cause an error when an attempt is made to execute it.

```
>>> 10 * (1/0)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ZeroDivisionError: division by zero
>>> 4 + spam*3
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'spam' is not defined
>>> '2' + 2
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: Can't convert 'int' object to str implicitly
```

Handling Exceptions

Even if a statement or expression is syntactically correct, it may cause an error when an attempt is made to execute it.

```
>>> while True:
...     try:
...         x = int(input("Please enter a number: "))
...         break
...     except ValueError:
...         print("Oops!  That was no valid number. Try again...")
```

THANK YOU