[Python List Comprehension]

by Alex Kelin				
prompt	command	result		
Concept	<pre>variable = [***result*** for element in entity if condition]</pre>	<pre>**entity could be range(), list, any iterable value</pre>		
Multiply each number	lib = [4,8,2,4,0,3] double_nums = [num * 2 for num in lib]	>>> print(double_nums) [8, 16, 4, 8, 0, 6]		
POW each number	lib = [4,8,2,4,0,3] pow_nums = [pow (x, 2) for x in lib]	>>> print(pow_nums) [16, 64, 4, 16, 0, 9]		
Reverse a list	<pre>one = ['a', 'b', 'c', 'd', 'e'] two = one[::-1] or three = ['a', 'b', 'c', 'd', 'e'][::-1]</pre>	>>> print(two) ['e', 'd', 'c', 'b', 'a'] >>> print(three) ['e', 'd', 'c', 'b', 'a']		
Traverse a list: every second value from indexes 1 to 6	<pre>one = ['a', 'b', 'c', 'd', 'e', 'f', 'g'] result = one[2:6:2] or result = [x for x in values[2:6:2]]</pre>	>>> print(result) ['c', 'e']		
Operations with strings	<pre>names = ['Bob', 'Mike', 'John'] new_list = ["Hi, " + name for name in names] or new_list = [f 'Hi, {name}' for name in names]</pre>	<pre>>>> print(new_list) ['Hi, Bob', 'Hi, Mike', 'Hi, John']</pre>		
String call of the first char	<pre>names = ['Bob', 'Mike', 'John', 'Jerry'] new_list = [x[0] for x in names]</pre>	>>> print(new_list) ['B', 'M', 'J', 'J']		
Length of a string	<pre>names = ['Bob', 'Mike', 'John', 'Jerry'] lengths = [len(x) for x in names]</pre>	>>> print(lengths) [3, 4, 4, 5]		
Unique values only	<pre>values = ['h',1,'b','b',4,'l','a',4] option_1 = list({x for x in values}) or option_2 = list(set(values)) or option_3 = [x for x in set(values)] or option_4 = [] [option_4.append(x) for x in values if x not in option_4]</pre>	<pre>>>> print(option_1) [1, 'h', 4, 'a', 'b', '1'] >>> print(option_2) [1, 'h', 4, 'a', 'b', '1'] >>> print(option_3) [1, 'h', 4, 'a', 'b', '1'] >>> print(option_4) ['h', 1, 'b', 4, '1', 'a']</pre>		
Common values	<pre>one = ['a', 1, 'b', 'b', 4, '1'] two = ['h', 'l', 1, 'a', 'j', 'l'] common = [x for x in one if x in two]</pre>	>>> print(common) ['a', 1, '1']		
Unite two lists	<pre>a = [5,1,6] b = [3,2,4] united = [x for y in [a, b] for x in y] or united = [x for x in a + b]</pre>	>>> print(united) [5, 1, 6, 3, 2, 4]		

```
one = ['Jack', 'Brit', 'Lucas', 'Ben']
                                                                                 >>> print(nl)
                       two = [10, 15, 4, 6]
                                                                                 [['Jack', 10], ['Brit',
Create nested list
                       nl = [[name, age] for name, age in zip(one, two)]
                                                                                 15], ['Lucas', 4],
                                                                                 ['Ben', 6]]
                       nl = [[one[i], two[i]] for i in range(len(one))]
                       nl = [[4, 8], [15, 2], [23, 42]]
                       sum = [x + y for x, y in nl]
                                                                                 >>> print(sum)
Nested list sum
                                                                                 [12, 31, 65]
                       or
                       sum = [x + y for (x, y) in nl]
                       nl = [[4, 8], [15, 2], [23, 42]]
                       check = [x > y for x, y in nl]
                                                                                 >>> print(check)
Nested list check
                                                                                 [False, True, False]
                       or
                       check = [x > y \text{ for } (x, y) \text{ in nl}]
                       a = [5, 1, 6]
                                                                                 >>> print(new)
Sum integers two
                       b = [3, 2, 4]
lists
                                                                                 [8, 3, 10]
                       new = [x + y \text{ for } x, y \text{ in } zip(a, b)]
Conditional
                       one = [1, 2, 3, 4, 5, 6, 7]
                                                                                 >>> print(new)
comprehension I,
                       \underline{\text{new}} = [x \text{ if } x \% 2 == 0 \text{ else } x * 2 \text{ for } x \text{ in one}]
                                                                                 [2, 2, 6, 4, 10, 6, 14]
ternery operator
                       a = [1, 2, 3, 4, 5, 6, 7, 8, 9]
                                                                                 >>> print(b)
Conditional
comprehension II
                       b = [x \text{ for } x \text{ in } a \text{ if } x > 5 \text{ and } x \% 2 == 0]
                                                                                 [6, 8]
                       sent = 'it is I, Kai, Jack, and Brit'
                                                                                 >>> print(c)
Multiple Condition
                       c = [x \text{ for } x \text{ in sent.split() if } x[0].isupper() and
comprehension I
                                                                                 ['Brit']
                       len(x) > 1 if ',' not in x]
                       all_clients = [{'name': 'Jack', 'age': 10,
                       'balance': 100}, {'name': 'Brit', 'age': 15,
                       'balance': 200}, {'name': 'Lucas', 'age': 4,
                                                                                 >>> print(checked)
Multiple Condition
                        'balance': 300}, {'name': 'Ben', 'age': 6,
comprehension II
                                                                                 ['Lucas', 'Ben']
                        'balance': 400}]
                       checked = [x['name'] for x in all_clients if
                       x['balance'] >= 300 \text{ or } x['age'] > 20]
                       booleans = [True, False, True]
                                                                                 >>> print(result)
Opposite boolean
                       result = [not x for x in booleans]
                                                                                 [False, True, False]
                                                                                 >>> print(check)
                       names = ['Bob', 'Mike', 'John', 'Jerry']
Check for value I
                                                                                 [False, False, True,
                       check = [x == 'John' for x in names]
                                                                                 False]
                       lib = [4, 8, 2, 4]
                                                                                 >>> print(check)
Check for value II
                       check = [x > 3 for x in lib]
                                                                                 [True, True, False, True]
                       names = ['Bob', 'Mike', 'John', 'Jerry', 'John']
                                                                                 >>> print(check)
Search for value
                       check = [i for i, x in enumerate(names) if x ==
index
                                                                                 [2, 4]
                        'John']
```

{ Python : Dictionary Comprehension }

	by Alex Kelin	
prompt	command	result
Concept	<pre>dictionary = { key: value for vars in iterable if condition}</pre>	While comprehension len(keys)=len(values)
Merge two lists	<pre>one = [1, 2, 3, 4, 5] two = ['a', 'b', 'c', 'd', 'e'] hm = dict([(one[i], two[i]) for i in range(len(one))]) or hm = {one[i]: two[i] for i in range(len(one))}</pre>	>>> print(hm) {1: 'a', 2: 'b', 3: 'c', 4: 'd', 5: 'e'}
Merge two lists with zip()	<pre>one = [1, 2, 3, 4, 5] two = ['a', 'b', 'c', 'd', 'e'] hm = {key: value for (key, value) in zip(one, two)} or hm = dict(zip(one, two)) or hm = {a: b for a, b in zip(one, two)}</pre>	>>> print(hm) {1: 'a', 2: 'b', 3: 'c', 4: 'd', 5: 'e'}
Merge dicts	<pre>one = {1: 'a', 2: 'b', 3: 'c'} two = {4: 'd', 5: 'e', 6: 'f'} united = {**one, **two} or one.update(two) or one.update(two) united = one</pre>	>>> print(united) {1: 'a', 2: 'b', 3: 'c', 4: 'd', 5: 'e', 6: 'f'} >>> print(one) {1: 'a', 2: 'b', 3: 'c', 4: 'd', 5: 'e', 6: 'f'} >>> print(united) {1: 'a', 2: 'b', 3: 'c', 4: 'd', 5: 'e', 6: 'f'}
Add values	<pre>one = {1: 'a', 2: 'b', 3: 'c'} one[4] = 'd' one[5] = 'e' one[6] = 'f' or one.update({4: 'd', 5: 'e', 6: 'f'})</pre>	>>> print(one) {1: 'a', 2: 'b', 3: 'c', 4: 'd', 5: 'e', 6: 'f'}
Moderate dict	<pre>old_stock = {'water': 1.42, 'cheese': 2.5, 'milk': 2.0} price = 0.76 correction = {item: value*price for (item, value) in old_stock.items()}</pre>	>>> print(correction) {'water': 1.0792, 'cheese': 1.9, 'milk': 1.52}
Unique values only (order preserved)	<pre>one = [4, 1, 2, 2, 3, 1] two = ['a', 'a', 'c', 'e'] new_two = [] uv = [new_two.append(i) for i in two if i not in new_two] result = {i: j for i, j in zip(one, new_two)}</pre>	>>> print(result) {4: 'a', 1: 'c', 2: 'e'}
Unique values only (order not preserved)	<pre>one = [4, 1, 2, 2, 3, 1] two = ['a', 'a', 'c', 'c', 'e'] result = {i: j for i, j in zip(one, set(two))}</pre>	>>> print(result) {4: 'c', 1: 'e', 2: 'a'}

```
a = [1, 2, 3, 4, 5]
                                                                                             >>> print(result)
Limited by values
                                                                                             {'a': 1, 'b': 2, 'c': 3}
length
                        result = \{k: v \text{ for } k, v \text{ in } zip(b, a[:len(b)])\}
                        names = {'mike': 10, 'jack': 32, 'rachel': 55}
                                                                                             >>> print(new_dict)
Dict with multiple
                        new dict = {k: v for (k, v) in names.items() if v % 2
conditions
                                                                                              {'jack': 32}
                        == 0 \text{ if } v > 20
Conditional
                        a = {'mike': 10, 'jack': 32, 'rachel': 55}
                                                                                             >>> print(a)
comprehension I
                        new dict = {k: v for (k, v) in a.items() if v % 2 ==
                                                                                              {'mike': 10, 'jack':
                        names = {'jack': 38, 'tina': 48, 'ron': 57, 'john':
                                                                                             >>> print(new dict)
                                                                                              {'jack': 'young',
'tina': 'old',
Conditional
                        new_dict = \{x: ('old' if y > 40 else 'young') for (x, young') \}
comprehension II,
ternery operator
                        y) in names.items()}
                                                                                              'Ron': 'old', 'john':
                                                                                              'young'}
                        names = ['alice', 'bob', 'kate', 'kimber']
                                                                                             >>> print(view)
                        size = [1, 2, 3, 4, 5, 6, 7]
                                                                                              {'Alice': ' 1 small',
'Bob': ' 2 small',
'Kate': ' 3 nice',
'Kimber': ' 4 nice'}
Conditional
                        view = {names[i].capitalize(): (f' {size[i]} nice' if
comprehension III,
ternery operator
                        i \ge 2 else f' \{size[i]\}  small') for i in
                        range(len(names))}
                                                                                             >>> print(res)
                                                                                             {'a': {1: 'a', 2: 'a', 3: 'a'}, 'b': {1: 'b', 2: 'b', 3: 'b'}, 'c': {1: 'c', 2: 'c', 3: 'c'}, 'd': {1: 'd', 2: 'd', 3: 'd'}}
                        keys =['a', 'b', 'c', 'd']
Nested dictionary
                        values = [1, 2, 3]
comprehension I
                        res = {k1: {k2: k1 for k2 in values} for k1 in keys}
                                                                                             >>> print(res)
                        keys =['a', 'b', 'c', 'd']
Nested dictionary
                                                                                             {'a': [1, 2, 3], 'b': [1, 2, 3], 'c': [1, 2, 3], 'd': [1, 2, 3]}
                        values = [1, 2, 3]
comprehension II
                        res = {k1: [x for x in values] for k1 in keys}
                                                                                             >>> print(counted)
                        names = ['Alex', 'Tom', 'Johnson', 'Bi', 'Foobar']
                                                                                              {'alex': 4, 'tom': 3,
'johnson': 7, 'bi': 2,
'foobar': 6}
Find length of
variable
                        counted = {x.lower(): len(x) for x in names if x}
```

Python lambda λ : functions

by Alex Kelin

by Alex Kelin			
prompt	command	result	
Concept	<pre>function_name = (lambda variable(s): result if condition else result_2)</pre>	Condition and ternery is optional	
Simple value operation	squared = lambda x: x ** 2	>>> print(squared(5)) 25	
Multiple value operation	<pre>value = lambda x, y: x + 2 - y</pre>	>>> print(value(2,1)) 3	
Check single value	<pre>check_num = (lambda x: f'{x} is greater than 5' if x > 5 else f'{x} is not greater than 5')</pre>	>>> print(check_num(6)) 6 is greater than 5	
Check multiple values	<pre>check_num = (lambda x, y: f'{x} and {y} are greater than 5' if x > 5 and y > 5 else f'{x} and {y} are not greater than 5')</pre>	>>> print(check_num(2,3)) 2 and 3 are not greater than 5	
Check for value	<pre>lib = ['a', 'b', 'c', 'd'] boolean = list(map(lambda x: x == 'b', lib))</pre>	>>> print(boolean) [False, True, False, False]	
Any() or all() value check	<pre>lst = [1, 2, 3, 4, 5] check_1 = any(map(lambda x: x % 2 == 0, lst)) check_2 = all(map(lambda x: x % 2 == 0, lst))</pre>	>>> print(check_1) True >>> print(check_2) False	
Operations with list	lib = [3, 1, 2] a = [lambda x=_: x + 1 for _ in lib] b = [(lambda x: x * 2)(_()) for _ in a]	>>> print(a) [<function <li="">stcomp>.<lambda> at 0x104761a80>,] >>> print(b) [8, 4, 6]</lambda></function>	
	<pre>lib = [3, 1, 2, 4] c = list(map(lambda x: x, lib)) d = list(map(lambda x: x / 2, lib))</pre>	>>> print(c) [3, 1, 2, 4] >>> print(d) [1.5, 0.5, 1.0, 2.0]	
	<pre>lib = ['Bob', 'Mike', 'John', 'Jerry'] e = list(map(lambda x: f' Hi, {x}', lib))</pre>	>>> print(e) ['Hi, Bob', 'Hi, Mike', 'Hi, John', 'Hi, Jerry']	
	<pre>lib_1 = ['a', 'b', 'c', 'd'] lib_2 = [20, 'M', 'T', 'V'] f = list(map(lambda x, y: f'{x} - {y}', lib_1, lib_2))</pre>	>>> print(f) ['a - 20', 'b - M', 'c - T', 'd - V']	
Operations with dict	<pre>a = ['a', 'b', 'c'] b = [1, 2, 3] new_dict = dict(zip(a, map(lambda x: x+1, b)))</pre>	>>> print(new_dict) {'a': 2, 'b': 3, 'c': 4}	
Determine length of a string	<pre>names = ['Bob', 'Mike', 'John', 'Jerry'] lengths = [len(x) for x in names]</pre>	>>> print(lengths) [3, 4, 4, 5]	
Opposite boolean	booleans = [True, False, True] result = [not x for x in booleans]	>>> print(result) [False, True, False]	
Extract positive values	<pre>my_list = [1, -2, 3, -4, 5] pos_nums = list(filter(lambda x: x > 0, my_list))</pre>	>>> print(pos_nums) [1, 3, 5]	