Python lambda λ : functions

by Alex Kelin

prompt	command	result
Concept	`function_name = ( lambda variable(s): result if condition else result_2 )`	Condition and ternery is optional
Simple value operation	`squared = lambda x: x ** 2`	`>>> print(squared(5))` `25`
Multiple value operation	`value = lambda x, y: x + 2 - y`	`>>> print(value(2,1))` `3`

Check single value	`check_num = (`
	`    lambda x: f'{x} is greater than 5' if x > 5`
	`    else f'{x} is not greater than 5')`
		`>>> print(check_num(6))`
		`6 is greater than 5`

Check multiple values	`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')`
		`>>> print(check_num(2,3))`
		`2 and 3 are not greater than 5`

Check for value	`lib = ['a', 'b', 'c', 'd']` `boolean = list(map(lambda x: x == 'b', lib))`	`>>> print(boolean)` `[False, True, False, False]`

Any() or all() value check	`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))`
		`>>> 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 <listcomp>.<lambda> at 0x104761a80>, ......]`
		`>>> print(b)`
		`[8, 4, 6]`

	`lib = [3, 1, 2, 4]`
	`c = list(map(lambda x: x, lib))`
	`d = list(map(lambda x: x / 2, lib))`
		`>>> print(c)`
		`[3, 1, 2, 4]`
		`>>> print(d)`
		`[1.5, 0.5, 1.0, 2.0]`

	`lib = ['Bob', 'Mike', 'John', 'Jerry']` `e = list(map(lambda x: f' Hi, {x}', lib))`	`>>> print(e)` `[' Hi, Bob', ' Hi, Mike', ' Hi, John', ' Hi, Jerry']`

	`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))`
		`>>> print(f)`
		`['a - 20', 'b - M', 'c - T', 'd - V']`

Operations with dict	`a = ['a', 'b', 'c']`
	`b = [1, 2, 3]`
	`new_dict = dict(zip(a, map(lambda x: x+1, b)))`
		`>>> print(new_dict)`
		`{'a': 2, 'b': 3, 'c': 4}`

Determine length of a string	`names = ['Bob', 'Mike', 'John', 'Jerry']` `lengths = [len(x) for x in names]`	`>>> 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	`my_list = [1, -2, 3, -4, 5]` `pos_nums = list(filter(lambda x: x > 0, my_list))`	`>>> print(pos_nums)` `[1, 3, 5]`