2. Functional Programming

29 November 2023 08:33

FUNCTIONAL PROGRAMMING

- Involves some sort of callback
- Constructs: map, filter, reduce, zip, lambda → not keywords; lambda → keyword: used to create anonymous functions
- Aims to reduce the number of lines as much as possible

MAP

map ([function], [iterable])

Takes each value from the iterable and passes that value through the specified function. The resulting values are stored in a lazy object of the class "map" (iterator object).

NOTE: Multiple arguments

You can specify multiple iterables in the map function, but the map function only iterates till the last element of the smaller iterable

```
list(map(lambda x: x+1,[1,2,3,4]))
[2, 3, 4, 5]
list(map(lambda x,y: x+y,[1,2,3,4],[2,3,4,5,6,7]))
[3, 5, 7, 9]
type((map(lambda x,y: x+y,[1,2,3,4],[2,3,4,5,6,7])))
<class 'map'>
```

When to use map: When number of elements in the output = no. of elements in input iterable

FILTER

filter ([function], [iterable])

- Checks for Boolean values
- Values that return True for the function are stored into an iterator

```
tuple(filter(lambda x:x[0]=="a", ["advaith","magic","cricket","arts"]))

('advaith', 'arts')
```

- If function is None, then it returns all values that are True

```
list(filter([1,2,4,-3,-5,0,""]))

Traceback (most recent call last):
  File "<pyshell#11>", line 1, in <module>
    list(filter([1,2,4,-3,-5,0,""]))
TypeError: filter expected 2 arguments, got 1
```

When to use filter: When no. of elements in output ≠ no. of elements in input

```
Traceback (most recent call last):
  File "<pyshell#11>", line 1, in <module>
    list(filter([1,2,4,-3,-5,0,""]))
TypeError: filter expected 2 arguments, got 1
list(filter(None,[1,2,4,-3,-5,0,""]))

[1, 2, 4, -3, -5]
```

no. of elements in output ≠ no. of elements in input

REDUCE

import functools

functools.reduce([function], [iterable], [initial value])

- Reduces an iterable to a single value
- Function takes two arguments; two elements are taken at once into the function which returns some value. This continues progressively till the end of the iterable
- Initial value (if specified) is taken as the first argument of the function before going through iterators

```
from functools import reduce
reduce(lambda x,y: x+y, [1,2,3,4,5,6,7,8,9,10])
55
reduce(lambda x,y: x+y, [1,2,3,4,5,6,7,8,9,10],100)
155

reduce(lambda x,y:x+y[0],["Advaith","S","Kumar"],"")
'ASK'
```

LAMBDA

- Used to create anonymous, inline functions

lambda var1, var2..: [return value from function without specifying "return"]

- You can also create named functions by assigning a lambda function as the value of some variable.