|  |  |  |
| --- | --- | --- |
| American University of SharjahCollege of Engineering Department of Computer Science & Engineering  P. O. Box 26666, Sharjah, UAE |  | **Lab Instructor:** Praveena Kolli  **Office:** EB2-126  **Phone**: 971-6-5152352  **e-mail**: pkolli@aus.edu  **Semester**: Spring 2020 |

**CMP 321 - Programming languages Laboratory**

**Lab 4 – Python Functions**

**Objectives**

* Make use of Python specific features
* Define functions and nested functions

**Please explore and make use of Python features where possible. Code that does not follow this note will be penalized.**

**Exercise 1**

1. Write an iterative function that takes three lists of strings as arguments (students’ name, address, and major) and returns a list of strings mixing these three fields, formatted as per the example below. Do not use zip() or similar. You may assume all three lists comprise the same number of elements.

Example: >>> fun( ["Ahmed", "John", "Zina"], ["Dubai", "Sharjah", "Al Ain"],

["COE", "CMP", "ELE"] )

['Ahmed from COE lives in Dubai', 'John from CMP lives in Sharjah',

'Mohd from ELE lives in Al Ain']

1. Write a recursive function that performs the same functionality. Do not use zip().
2. Write a function that uses zip() to perform the same functionality.

**Exercise 2**

1. Write a function enum() that returns a list of tuples aggregating the given list elements with a counter (starting at 0). Do not use enumerate() or similar.

Example: >>> enum( ['A', 'B', 'C'] )

[ (0, 'A'), (1, 'B'), (2, 'C') ]

1. Modify enum() to take a default starting value and default step for the counter.

Examples: >>> enum( ['A', 'B', 'C'], 5 ) >>> enum( ['A', 'B', 'C'], 4, 2 )

[ (5, 'A'), (6, 'B'), (7, 'C') ] [ (4, 'A'), (6, 'B'), (8, 'C') ]

**Exercise 3**

1. Write a recursive function that returns true if an object is a member of a given list, or false otherwise. Hint: You should use slicing…

Examples: >>> member( 2, [1, 3, 5] ) >>> member( 4, [1, 2, 3, 4, 5] )

False True

1. Modify member() to return the index where the object is found, or -1 otherwise.

Examples: >>> member( 2, [1, 3, 5] ) >>> member( 4, [1, 2, 3, 4, 5] )

-1 3

**Exercise 4**

1. Write a recursive function apply\_to\_all() that takes a function and a list as arguments and returns a list comprising the results of applying the function to each elements of the list.

Example: >>> apply\_to\_all( sqrt, [4, 9, 16, 25] ) # sqrt = square root function

[2, 3, 4, 5]

1. Write a recursive function apply\_select() that takes a predicate function and a list as arguments and returns a list of those elements for which the predicate is true.

Example: >>> apply\_select( is\_odd, [2, 3, 5, 8, 9] ) # is\_odd = True is num is odd

[3, 5, 9] # False otherwise

**Exercise 5**

1. Write an accumulator recursive factorial function, with a helper function that is nested within. (Do not use default parameters.) Add “type checking” so that only int args work.
2. Write a combination function C(n,k). Write a function T(n) that returns a function of one argument k that computes C(n,k).

Example: >>> f = T(10) ; f(3) # f computes C(10,k)

120

Or

print(T(10)(3))

120