# Worksheet #1

***Dictionaries***

1. A snack bar has the following price:

|  |  |
| --- | --- |
| **Product** | **Price** |
| Salted | €0.60 |
| Snack | €1.20 |
| Juice | €1.50 |
| Refrigerator | €1.00 |
| Cake | €0.70 |

* 1. Create a dictionary to store the snack bar price list.
  2. Using the dictionary created in the previous paragraph, calculate what the price would be to pay if the customer ordered a snack, a cake and an orange juice.
  3. Develop a program that:
* question the customer's request (until the customer inputs "stop")
* Calculate the total order value
* ask how much the customer will pay
* calculate the change to return.

Example:

|  |  |
| --- | --- |
| **Input:** | **Output:** |
|  | Introduce products. To stop: stop |
| snack  cake  stop | Total: €1.9  how much do you pay? |
| 5 | Change = 3.1€ |

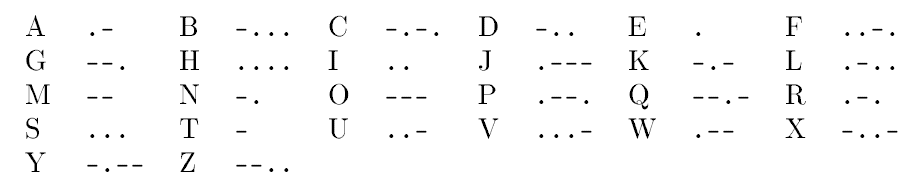
* 1. Create a function that uses the dictionary to average product prices.

1. Write a translate function which takes a list of words and a dictionary as arguments and returns a new list of words translated using the dictionary translation. If the word list contains a word that does not exist in the dictionary, that word must remain untranslated. The output must be displayed as a complete sentence [[1]](#footnote-1).

Example:

|  |  |
| --- | --- |
| **Function call:** | **Output:** |
| pt\_en ={ "today":" today ",  "this": " is ",  "misty":" cloudy "}  txt = [" today","this","very","misty "]  translate ( txt, pt\_en ) | today it is very cloudy |

1. Morse code associates each letter of the alphabet with a sequence of “dots” and “dashes”



* 1. Define the Morse code table as the following dictionary:

code = {"A": ".-", "B": "-...", "C": "-.-", "D": "-..", "E": "." , "F G H I J": ".-- --","K": "-.-","L": ".-..", "M": "--", "N": "-.", "O": "-- --", "P": ".--.", "Q": "--.-", "R": ".-.", "S": "...", "T": "-", "U": "..-","V": "-.---", "Z": "--.."}

* 1. morse( txt ) function that converts the letters in a sequence of characters to Morse; the result should be a string with dots and dashes. Use a space to separate strings corresponding to the letters. Characters in the original text that are not capital letters should be ignored.

Examples:

|  |  |
| --- | --- |
| **Function call:** | **Output:** |
| morse("ABC") | .- -... -.- |
| morse( "AB C") | .- -... -.- |
| morse( "ABC xyz ") | .- -... -.- |
| morse( "ATTACK AT DAWN") | .- - - .- -.- -.- .- - -.. .- -. |

1. In the national football championship, a victory counts 3 points, a draw 1 point and a loss 0 points. In each game, the team that scores the most goals wins, with a tie if the number of goals is the same.
   1. Implement the function football(scores) that returns a dictionary with the score of each team at the end of the season. The scores parameter is a list of dictionaries with the results of the journey. Each dictionary has a club's name in the bracket and the number of goals in value. For example,

Example:

|  |  |
| --- | --- |
| **Function call:** | **Output:** |
| soccer ([  {"Vitória SC":2, "Boavista":1},  {"Gil Vicente":1, "Rio Ave":1},  {"Famalicão":3, "Sporting":2},  {"FC Porto":0, "Benfica":0},  {"Tondela":2, "Santa Clara":3}  ]) | {  'Victory SC': 3,  'Boavista': 0,  'Gil Vicente': 1,  'Rio Ave': 1,  'Famalicão': 3,  'Sporting': 0,  'FC Porto': 1,  'Benfica': 1,  'Santa Clara': 3,  'Tondela': 0  } |

* 1. Add the following code to the developed one and check what happens

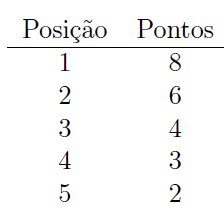
res = football (scores)

import operator

res = dict ( sorted ( res.items (), key= operator.itemgetter (1), reverse=True))

print( res )

1. In the 50s of the last century, in the Formula 1 championship, the score for each race was assigned based on the position of each driver at the end, according to the following table:



* 1. Implement the function formula1(scores) which, given a list with the ordered list of the five best ranked, for each event of a season, returns a dictionary with the score of each runner. For example,

Example:

|  |  |
| --- | --- |
| **Function call:** | **Output:** |
| formula1([  [' Sainz ', 'Verstappen', 'Hamilton', 'Ricciardo', 'Massa'],  ['Bottas', 'Verstappen', ' Raikkoten ', 'Stroll', 'Vettel'],  ['Perez', ' Raikkoten ', 'Verstappen', 'Hamilton', 'Vettel']  ]) | {  ' Sainz ': 8,  'Verstappen': 16,  'Hamilton': 7,  'Ricciardo': 3,  'Mass': 2,  'Boots': 8,  ' Raikkoten ': 10,  'Stroll': 3,  'Vettel': 4,  'Perez': 8  } |

* 1. Add code from exercise 4.2 and check what is changed.

1. This exercise aims to simulate the inventory of a computer store.
   1. Using dictionaries and lists, build the store's inventory. Inventory must contain at least 10 products. The information to include for each product is: product name, quantity and price.
   2. Implement the purchase function ( shoppinglist ) that

* receives a dictionary with the list of purchases (like: {product1: quantity1, …})
* if the products exist in the inventory and there is enough quantity for sale, they determine the final price of the products
* if they do not exist, or the stock is insufficient, show a message accordingly
* determine the total purchase price

Output example:

Insufficient stock of the product \*wireless mouse\*

price (3x keyboard [7.5€/ pc ]):22.5€

Product \*fries\* does not exist

Total purchase: 22.5€

1. <https://www.w3schools.com/python/ref_string_join.asp> [↑](#footnote-ref-1)