* A tener en cuenta
  + Cuando se imprime algo se puede usar: print('${} pesos mexicanos son ${} pesos colombianos'.format(ammount,result))
  + No se puede modificar un string para cambiar algo del string se crea uno nuevo asi: r = ‘l’ + s[1:]
  + Se puede “dividir un string” creando uno nuevo s[1:3] (va hasta el índice 3 pero no lo incluye), s[1:], s[1:6:2] (del índice uno al dos saltando de dos caractares), s[::-1] (voltea la palabra), en otras palablas s[inicio,final,saltos]
  + El range también puede ser range(inicio,final,saltos) y tampoco incluye el final, ej: range(1,10,2) -> 1,3,5,7,9
  + Generar números aleatorios: import random y luego random.randint(0,20)
  +  upper -> regresa un string en mayúsculas, lower -> regresa un string en minúsculas, find -> regresa el índice lógico de un carácter (nombre.find(‘L’)), Split -> separa con el carácter definido por el usuario.
* Listas
  + Para crear listas
    - Amigos = list()
    - Amigos = []
  + Añadir un objeto al final de la lista
    - Amigos.append(‘pedro’)
  + Eliminar el último elemento de la lista
    - Amigos.pop()
  + Ordenar la lista
    - Lista.sort()
  + Eliminar un elemento de la lista con su índice
    - del lista[índice]
  + Operaciones con listas
    - Suma -> lista3 = lista1 + lista2 -> une las dos listas
    - Multiplicación -> lista3 = lista1 \* 5 -> multiplica cada elemento dentro de la lista
    - Se puede usar slices lista3[::-1] y los demás
  + insert: añade un nuevo elemento a la lista
  + Convertir un String a lista
    - nuevaLista = list(Palabrastring)
  + Convertir una lista a string
    - nuevoString = ‘ ’.join(nuevaLista)
  + Buscar un caracter dentro de la lista:
    - Lista.index(‘caracteraBuscar’) -> el caso de otros lenguajes retorna un -1 si no lo encontró, pero en Python si no lo encuentra regresa un error y hay que usar try.
* Diccionarios: como en las listas se puede definir los diccionarios de la siguiente forma {} o también con la palabra dict(). Para añadir un elemento a un diccionario se hace: mi\_diccionario[‘primerElemento’] = ‘Hola’ ->aquí la llave es primerElemento de esta manera para acceder al elemento en el diccionario se hace: mi\_diccionario[‘primerElemento’]. Ej: KEYS = {‘a’:’w’,’b’:’E’}
  + Formas de iterar en un diccionario:
    - For llave in diccionario: ->retorna las llaves del diccionario
    - For llave in diccionario.keys(): igual que el anterior
    - For llave in diccionario.values(): ->retorna los valores que se asignaron
    - For llave, valor in diccionario.items(): print(llave,valor) -> muestra los dos valores
* Tuplas: se pueden definir de dos formas mi\_tupla = 1,2,3,4 o también mi\_tupla = (1,2,3,4). Las tuplas no pueden modificarse toca generar una tupla nueva. Para generar una tupla con un solo elemento la syntasis es: mi\_tupla = (1,) , sin la coma se vuelve un entero.
* Sets: son como las listas pero los elementos no se repiten, se parecen a teoría de conjuntos. s = set([1,2,3]) t = set([3,4,5])
  + Operaciones
    - Unión -> s.union(t) ->set([1,2,3,4,5])
    - Intersección -> s.intersection(t) ->set([3])
    - Diferencia -> s.difference(t) ->set([1,2])
  + Buscar datos:
    - 1 in s -> True
* Dictionary comprehension - list comprehension (azúcar sintáctica)
  + List comprehension: se puede reducir código así: even = [num for num in range(1,31) if num % 2 == 0]
  + Dictionary comprehension: squares = {num: num\*\*2 for num in range(1,11)}