Guia de Ejercicios de ADP, Clase 2. UNSAM 2018

A. Debugguear

Ejercicio 1:

La función *saludar()* pide nombre y apellido, los imprime en un mensaje de bienvenida y los devuelve. Sin embargo, al ejecutarlo da error. Hallar el error y corregirlo.

In [ ]:
```
def saludar():
    N=[]
    N.append(input("Nombre: "))
    N.append(input("Apellido: "))
    print("Bienvenido {0} {1}!".format(N[1],N[2]))
    return N

saludar()
```

Ejercicio 2:

En este fragmento ocurre algo extraño. Una variable cambia de valor cuando no pareciera tener que cambiar. Determine dónde y por qué cambia y corrija el código de forma conveniente.

In [ ]:
```
import numpy as np

def cuadrados(V):
    """
    esta función toma un array V
    e imprime sus elementos al cuadrado.
    """
    W=V
    n=len(V)
    for i in range(n):
        W[i]=V[i]**2
    print("El vector cuadrado: ",W)
    return W


V=np.linspace(0,5,6)
print("El vector original: ",V)
W=cuadrados(V)
print("El vector original: ",V)
print("El vector cuadrado: ",W)
```

Ejercicio 3:

Determinar por qué el gráfico de la parábola que realiza el siguiente código no se ve simétrico y corregirlo.

ojo: la priemra linea (*%matplotlib inline*) solo se usa si se corre en una notebook para que plotee inline, si lo corrés de otra forma hay que sacarla

In [ ]:
```
%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np

def F(x):
    return x**2

X = np.arange(-10,10)
plt.plot(X,F(X))
```

B. Estructuras de datos: Diccionarios, conjuntos y arrays

Diccionarios

Ejercicio 1:

Crear un diccionario que se llame precios. Agregar los siguientes elementos al diccionario:

```
'banana': 30
'manzana': 30
'naranja': 20
'pera': 30
```

Crear un diccionario que se llame stock, Agregarlos siguientes elementos al diccionario:

```
'banana': 6,
'manzana': 0,
'naranja': 32,
'pera': 15
```

1. Utilizando ciclos sobre los diccionarios correspondientes:
2. Reducir todos los precios un 10%.
3. Imprimir cada el nombre el precio y el stock de cada fruta. Por ejemplo, se espera que para la manzana se imprima:

`python manzana precio: 27.0 stock: 0`

2. Calcular cuánto dinero obtendría si logra vender toda la fruta al precio dado.

In [ ]:

Ejercicio 2:

Dado el siguiente diccionario

```
inventario = {
'cajon': ['libro', 'cordel', 'pelusa']
'mochila': ['libro', 'comida', 'plata', 'moneda'],
'bolsa': ['cordel', 'piedra', 'comida']
}
```

1. Agregar un item al diccionario cuya clave sea *'bolsillo'* y su valor sera una lista de las siguientes cadenas *'pelusa'*, *'fósforo usado'* y *'moneda'*.
2. Ordenar los items de la lista que tiene como clave *'mochila'*.
3. Luego, remover *'comida'* de la lista de items de la clave *'mochila'*.

In [ ]:

SET

Ejercicio 3:

1. Basándose en el diccionario del ejercicio anterior crée un conjunto (*set()*) que contenga los elementos que se encuentran en el *cajon*, la *mochila*, la *bolsa* y el *bolsillo*
2. Agregue al conjunto el elemento *'cortaplumas'*.
3. Elimine del conjunto la *moneda*.
4. Calcule la cantidad de elementos que hay en el conjunto.

In [ ]:

NUMPY ARRAYS

Ejercicio 4:

1. Crear un arreglo de Numpy Array de 3x3 que tenga todos los valores de 2 a 10 (inclusive).

Output esperado: `python [[ 2 3 4] [ 5 6 7] [ 8 9 10]]`

2. Crear un Numpy Array de ceros de longitud 10. Luego actualice el sexto valor a 11. Output esperado 1: `[ 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]`

Output esperado luego de actualizar el sexto lugar: `[ 0. 0. 0. 0. 0. 0. 11. 0. 0. 0.]`

3. Crear un Numpy Array que contenga todos los valores de 12 a 38.

Output esperado: `[12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37]`

In [ ]:

Ejercicio 5:

Realizar un programa en Python que tenga como input un Numpy Array de valores de temperatura en grados Celcius y dé como output un Numpy Array de temperaturas en Fahrenheit.

In [ ]:

Ejercicio 6:

Utilizando la función *numpy.where()* o *numpy.nonzero()*, escribir un par de líneas de código en python que, dado un np.array *x*, devuelva los elementos de *x* mayores que 10 y sus coordenadas.

Ejemplo:

```
Input: [[ 0 10 20], [20 30 40]]

Valores mayores a 10 = [20 20 30 40]
Indices:  (array([0, 1, 1, 1]), array([2, 0, 1, 2]))
```

In [ ]:

C. Análisis de código

Ejercicio:

El siguiente fragmento de código realiza un *experimento numérico*. Analizarlo para describir qué hace cada una de las tres funciones. ¿Qué conclusión puede sacar de los resultados del experimento?

In [8]:
```
import numpy as np
from numpy.random import randint as randint

def busqueda_alea(n):
    cota_inf = 1
    cota_sup = 1024
    hallado = False
    n_intentos=0
    while (not hallado):
        n_intentos += 1
        propuesta = randint(cota_inf,cota_sup)
        #print(n_intentos, propuesta, cota_inf, cota_sup)
        if (propuesta == n):
            hallado = True
        elif (propuesta < n):
            cota_inf = propuesta
        else: #en este caso propuesta > n
            cota_sup = propuesta
    return n_intentos

def busqueda_bina(n):
    cota_inf = 1
    cota_sup = 1024
    hallado = False
    n_intentos=0
    while (not hallado):
        n_intentos += 1
        propuesta = (cota_inf + cota_sup) // 2
        #print(n_intentos, propuesta, cota_inf, cota_sup)
        if (propuesta == n):
            hallado = True
        elif (propuesta < n):
            cota_inf = propuesta
        else: #en este caso propuesta > n
            cota_sup = propuesta
    return n_intentos

def comparar():
    intentos_alea = []
    intentos_bina = []

    for i in range(10000):
        n=randint(1,1024)
        intentos_alea.append(busqueda_alea(n))
        intentos_bina.append(busqueda_bina(n))

    print("Observar que {0} es menor que {1}".format(np.mean(intentos_bina), np.mean(intentos_alea)))

comparar()
```

```
Observar que 9.007 es menor que 13.0865
```