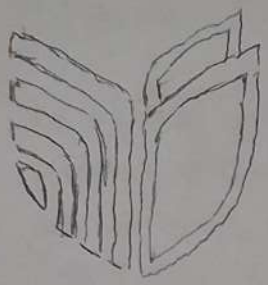


Instituto Tecnológico Superior de
San Andrés Tuxtla

ALUMNO: Karla Guadalupe Quino Cinta

MATERIA: Programación para Ciencia de datos

DOCENTE: Rogelio Enrique Telona Torres

GRUPO: 810-A

Introducción

En python, existen estructuras de datos como las listas y los diccionarios. Las listas permiten almacenar colecciones ordenadas de elementos, mientras que, los diccionarios son colecciones no ordenadas de pares clave-valor.
Este trabajo se centrará en tres aspectos: las operaciones que no modifican las listas y diccionarios, las operaciones que sí lo hacen y sobre las copias de ambas estructuras.

Operaciones que no modifican una lista

- len(l): Devuelve el número de elementos de la lista l.
- min(l): Devuelve el mínimo elemento de la lista siempre que los datos sean comparables
- max(l): Devuelve el máximo elemento de la lista l siempre que los datos sean comparables
- sum(l): Devuelve la suma de los elementos de la lista l, siempre que los datos se puedan sumar.
- dato in l: Devuelve True si el dato pertenece a la lista l y False en caso contrario
- l.index(dato): Devuelve la posición que ocupa en la lista el primer elemento con valor dato.
- l.count(dato): Devuelve el número de veces que el valor *dato* está contenido en la lista l
- all(l): Devuelve True si todos los elementos de la lista l son True y false en caso contrario
- any(l): Devuelve True si algún elemento de la lista l es True y False en caso contrario.

Operaciones que modifican una lista

- l1 + l2: Crea una nueva lista concatenan los elementos de las listas l1 y l2.
- l.append(dato): Añade *dato* al final de la lista l.
- l.extend(sequencia): Añade los datos de *sequencia* al final de la lista l.
- l.insert(indice, dato): Inserta *dato* en la posición *indice* de la lista l y desplaza los elementos una posición a partir de la posición *indice*.

- l.remove(dato): Elimina el primer elemento con valor *dato* de la lista l y desplaza los que están por detrás de él una posición hacia adelante.
- l.pop([indice]): Devuelve el dato en la posición *indice* y lo elimina de la lista l, desplazando los elementos por detrás de él una posición hacia adelante.
- l.sort(): Ordena los elementos de la lista l de acuerdo al orden predefinido, siempre que los elementos sean comparables.
- l.reverse(): invierte el orden de los elementos de la lista l.

Copia de listas

Existen dos formas de copias listas:

- Copia por referencia `l1 = l2`: Asocia la variable l2 la misma lista que tiene asociada la variable l2, es decir, ambas variables apuntan a la misma dirección de memoria. Cualquier cambio que hagamos a través de l1 o l2 afectará a la misma lista.
- Copia por valor `l1 = list(l2)`: Crea una copia de la lista asociada a l2 en una dirección de memoria diferente y se la asocia a l1. La variables apuntan a direcciones de memoria diferentes que contienen los mismos datos. Cualquier cambio que hagamos a través de l2 no afectará a la lista de l2 y viceversa.

Operaciones que no modifican un diccionario

- len(d): Delvuelve el número de elementos del diccionario d.
- min(d): Devuelve la mínima clave del diccionario d siempre que las claves sean comparables
- max(d): Devuelve la máxima clave del diccionario d siempre que las claves sean comparables.
- sum(d): Devuelve la suma de las claves del diccionario d, siempre que las claves se puedan sumar
- clave in d: Devuelve True si la clave *clave* pertenece al diccionario d y False en caso contrario.
- d.keys(): Devuelve un iterador sobre las claves de un diccionario
- d.values(): Devuelve un iterador sobre los valores de un diccionario.

- d.items(): Devuelve un iterador sobre los pares clave-valor de un diccionario.

Operaciones que modifican un diccionario

- d[clave]=valor: Añade al diccionario d el par formado por la clave clave y el valor valor
- d.update(d2): Añade los pares del diccionario d2 al diccionario d.
- d.pop(clave, alternativo). Devuelve el valor asociado a la clave clave del diccionario d y lo elimina del diccionario. Si la clave no está, devuelve el valor alternativo.
- d.popitem(). Devuelve la tupla formada por la clave y el valor del último par añadido al diccionario d y lo elimina del diccionario.
- del d[clave]. Elimina del diccionario d el par con la clave clave
- d.clear(). Elimina todos los pares del diccionario d de manera que se queda vacío.

Copia de diccionarios.

- Copia por referencia d1=d2: Asociada a la variable d1 el mismo diccionario que tiene asociado la variable d2, es decir, ambas variables apuntan a la misma dirección de memoria. cualquier cambio que hagamos a través de d1 o d2 afectará el mismo diccionario.
- Copia por valor d1=list(d2). Crea una copia del diccionario asociado a d2 en una dirección de memoria diferente y se le asocia a d1. Las variables apuntan a direcciones de me memoria diferentes que contienen los mismos datos. Cualquier cambio que hagamos a través de d1 no afectará a d2 y viceversa.

Conclusiones

Se han explorado las diferentes que se pueden realizar sobre las listas y los diccionarios en Python. Es importante identificar cada una de ellas, ya que, por ejemplo al utilizar las operaciones que no modifican la estructura pueden ser útiles para realizar operaciones sin afectar el contenido de la lista o diccionario.

"Listas | aprende con alf". Aprende con Alf. Accedido el 12 de febrero de 2025 [En línea]. Disponible: https://aprendeconalf.es/docencia/python/manual/listas/

Valor = 20

Curso: Programación para Ciencia de Datos

Tarea: Investigación Unidad 1

Ver todos los envíos

Karla Guadalupe Quino Cinta

krlcinya23@gmail.com

Fecha de entrega: 16 de febrero ...

Cambiar usuario

2 de 8 Reiniciar preferencias de tabla

Página 1 de 4

Entrega

Enviado para calificar

Calificado

La tarea fue enviada 3 días 19 horas antes de la fecha límite

Los estudiantes pueden editar este envío

INV_PCD_U1.pdf 12 de febrero de 2025, 20:18

Comentarios (0)

Calificación

Calificación:

Hoja de presentación	No contien todos los datos ***0 puntos***	Datos incompletos ***0.5 puntos***	Completo ***1 puntos***	
Introducción	No contiene ***0 puntos***	Muy pequeña ***1.5 puntos***	Completa ***3 puntos***	
Contenido	No cubre los temas ***0 puntos***	La mitad de los temas ***6 puntos***	Completo ***11 puntos***	
Referencias IEEE	No contiene ***0 puntos***	Una o no tiene el formato ***1 puntos***	Más de una y formato correcto ***2 puntos***	
Conclusión	No contien ***0 puntos***	Muy pequeña ***1 puntos***	Completa ***2 puntos***	
Archivo PDF	Sin formato ***0 puntos***	Correcto ***1 puntos***		

Calificación actual en el libro
20.00

Comentarios de retroalimentación

Notificar a estudiantes

GUARDAR CAMBIOS GUARDAR Y MOSTRAR SIGUIENTE REINICIAR

TEC

Rogelio Enrique Telona Torres

UNIDAD 1

introduccion a python

Programacion para ciencia de datos

5 de febrero de 2025

8vo semestre

Ingenieria informatica

esta es una introduccion a python , iniciaremos desde 0.

salida de datos por pantalla , para eso emplearemos la funcion print()

In [ ]:
```
print("HOLA MUNDO")
```

In [1]:
```
print("      +")
print("      *")
print("     * *")
print("    * * * ")
print("  * * * * * ")
print(" * * * * * * * ")
print("    |   |   ")
```

```
     +
     *
    * *
   * * *
 * * * * *
* * * * * *
   |   |
```

OPERACIONES ARTIMETICAS

+ suma

- resta

* producto

/ division

** potencia

// division entera

In [2]:
```
# no es necesrio indocar el tipo de dato
a = 3
b = 2
print ("suma = ", a + b)
print ("resta = ", a - b)
print ("producto = ", a * b)
print ("division = ", a / b)
print ("potencia = ", a ** b)
print ("suma = ", a // b)
```

```
suma =  5
resta =  1
producto =  6
division =  1.5
potencia =  9
suma =  1
```

prioridad de los operadores

1. potencia y radicales

2. multiplicacion, division y residuo

3. suma y resta

In [4]:
```
# esto es un ejemplo

print (((2+3) * (5**2+5))/2)
```

```
75.0
```

type() saber el tipo de dato

In [8]:
```
x = (((2+3) * (5**2+5))/2)
type(x)
y= 9
type(y)
```

Out[8]: int

La lectura de datos por teclado, se realiza usando la funcion **input()**, la cual lee lo que el usuario escriba

inputraw() - version 2 de python

NOTA: todo lo que se le es es **texto o caracter**

In [3]:
```
print("escribe algo")
texto = input()
```

```
print(texto)
```

```
escribe algo
ola
```

Ejercicio 1

suma de dos numero ingresados por teclado

In [10]:
```
print("suma de dos numeros")

print("ingrese primero numero")
numero1 = int(input())
print("Ingrese el segundo numero")
numero2 = int(input())
suma = numero1 + numero2
print(numero1 , "+" , numero2,  "= ",suma)
```

```
suma de dos numeros
ingrese primero numero
Ingrese el segundo numero
3 + 6 =  9
```

Mismo problema , pero **CUQUIS**

In [3]:
```
number1= int(input( "Escribe un numero"))
number2= int(input( "Escribe un numero"))
add = number2 + number1
print(number1, " + ", number2, " = ", add)
print(f'{number1} + {number2} = {add}')
print('{} + {} = {}'.format(number1,number2,add))
```

```
3  +  6  =  9
3 + 6 = 9
3 + 6 = 9
```

Operaciones condicionales

== comparacion

!= diferente o no es igual

> mayor

< menor

>= mayor o igual

<= menor o igual

In [ ]:
```
a =33
b=200
```

```
if a > b:
    print(f"{a} es mayor que {b}")

else:
    print(f"{b} es mayor que {a}")
```

```
200 es mayor que 33
```

In [7]:
```
a =200
b=200

if a > b:
    print(f"{a} es mayor que {b}")

elif b>a:
    print(f"{b} es mayor que {a}")

else:
    print(f"{b} es igual que {a}")
```

```
200 es igual que 200
```

crear un script en python que lea 4 numero enteros, debe mostrar en pantalla la suma del primer y ultimo numero y el producto del segundo y tercer numero leido

In [12]:
```
number1= int(input( "Escribe el primer numero"))
number2= int(input( "Escribe el segundo numero"))
number3= int(input( "Escribe el tercer numero"))
number4= int(input( "Escribe el cuarto numero"))

print("primer numero:", number1, "Segundo numero: " , number2, "tercer numero: " ,
print(f'{number1} + {number4} = {number1 + number4}')
print(f'{number2} * {number3} = {number2 * number3}')
```

```
primer numero: 5 Segundo numero:  6 tercer numero:  4 Cuarto numero:  9
5 + 9 = 14
6 * 4 = 24
```

Ejercicio con if else elif

La pizzería "Italiana" ofrece pizzas vegetarianas y no vegetarianas a sus clientes. Los ingredientes para cada tipo de pizza aparecen a continuación.

Ingredientes vegetarianos: Pimiento y tofu.

Ingredientes no vegetarianos: Peperoni, Jamón y Salmón.

Escribir un programa que pregunte al usuario si quiere una pizza vegetariana o no, y en función de su respuesta le muestre un menú con los ingredientes disponibles para que elija. Solo se puede eligir un ingrediente además de la mozzarella y el tomate que están en todas la pizzas. Al final se debe mostrar por pantalla si la pizza elegida es vegetariana o no y todos los ingredientes que lleva

In [ ]:
```
print("      ====================================")
print ("     || BIENVENIDO A LA PIZZERIA ITALIANA  ||")
print("      ====================================")
```

```
elegir_pizza = input("\n¿Quieres una pizza vegetariana?(s/n) ")

if elegir_pizza == 's':
    print("Ingredientes disponibles para la pizza vegetariana: \n1. Pimientos\n2. T

    elegir_ingrediente = input("\n¿Qué ingrediente quiere? Elija un número: ")

    if elegir_ingrediente == '1':
        print(f"\nSu pizza es vegetariana y tiene los siguientes ingredintes:\n Moz
    elif elegir_ingrediente == '2':
        print(f"\nSu pizza es vegetariana y tiene los siguientes ingredeintes:\n Mo
    else:
        print("Opción no válida, por favor elija '1' o '2' ")

elif elegir_pizza == 'n':
    print("Ingredientes disponibles para la pizza no vegetariana: \n1. Peperoni\n2.

    elegir_ingrediente = input("\n¿Qué ingrediente quiere? Elija un número: ")

    if elegir_ingrediente == '1':
        print(f"\nSu pizza es no vegetariana y tiene los siguientes ingredeintes:\n
    elif elegir_ingrediente == '2':
        print(f"\nSu pizza es no vegetariana y tiene los siguientes ingredeintes:\n
    elif elegir_ingrediente == '3':
        print(f"\nSu pizza es no vegetariana y tiene los siguientes ingredeintes:\n
    else:
        print("Opción no válida, por favor elija '1', '2' o '3'")

else:
    print("Opción no válida, por favor ingrese 's' o 'n' ")
```

```
 ====================================
|| BIENVENIDO A LA PIZZERIA ITALIANA  ||
 ====================================
Ingredientes disponibles para la pizza vegetariana:
1. Pimientos
2. Tofu

Su pizza es vegetariana y tiene los siguientes ingredeintes:
 Mozzarella, Tomate y Tofu
```

Ciclos

While

For

In [1]:
```
i = 0
while i<6:
    print(i)
    i+=1
```

```
0
1
2
3
4
5
```

In [ ]:
```
for x in range(6):  #por default inincia en 0
    print(x)
```

```
0
1
2
3
4
5
```

In [ ]:
```
for a in range (1,6):  #el par 1,6 corresponde al inicio  N-1
    print(a)
```

```
1
2
3
4
5
```

In [ ]:
```
for y in range(1,11,2):    #inicio, fin, incremento
    print(y)
```

```
1
3
5
7
9
```

In [12]:
```
frase = "paiton"
for leter in frase:
    print(leter)
    #print(leter,end="")   #elimina el salto de linea
```

```
p
a
i
t
o
n
```

Listas

En las listas podemos ingresar todo tipo de datos

se utilza [] para definirlas y los elementos se separan usando comas

las listas son elementos mutables (pueden cambiar sus variables)

In [ ]:
```
lista = [11,3,4,5]
print(lista)
print(lista[3])  #indice inicia 0 .... desde la derecha
print(lista[-2]) #indice inicia -1 .... desde la derecha
```

```
[11, 3, 4, 5]
5
4
```

In [18]:
```
for elemento in lista:
    print(elemento)
```

```
11
3
4
5
```

In [15]:
```
palabra = "paiton"
print(palabra[3])
print(palabra[-1])
```

```
t
n
```

In [16]:
```
# palabra = paiton
# SLICINGM seleccion de varios elementos [inicio:final-1]
print(palabra[3:6])    #subtraer una cadena
print(palabra[:2])
print(palabra[2:])
print(palabra[1:6:2])
print(palabra[1:6:3])
```

```
ton
pa
iton
atn
ao
```

In [ ]:
```
numeros = [1,2,3,4,5,6,7,8,9,10]
copia = numeros
#type(numeros)
#type(copia)
print(numeros)
numeros[5] = 'elsa'
print(numeros)
# las listas no se copian al asignaralas a otra variable
# es un puntero a la lista original
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
[1, 2, 3, 4, 5, 'elsa', 7, 8, 9, 10]
```

In [7]:
```
orginal = [1,3,5,7,9,11,13,15]

cpp = orginal[:]    #copia explicita
```

```
print(f'lista original :{orginal}')
print(f'lista original :{cpp}')

orginal[4] = 10
print("original modificada: ", orginal)
print("copia modificada:    ", cpp)
```

```
lista original :[1, 3, 5, 7, 9, 11, 13, 15]
lista original :[1, 3, 5, 7, 9, 11, 13, 15]
original modificada:  [1, 3, 5, 7, 10, 11, 13, 15]
copia modificada:     [1, 3, 5, 7, 9, 11, 13, 15]
```

INDEXACIÓN ANIDADA

In [ ]:
```
lespecial = [1,2,3,[4,5,['tercera', 'valueError']]]
print(lespecial[0])
print(lespecial[3]) # accede a la sublista
print(lespecial[3][2]) #accede al segundo elemento de la sublista
print(lespecial[3][2][1]) # accede al elemento 1 de la sublista
print(lespecial[3][2][1][5])
print(lespecial[3][2][1][3:])
print(lespecial[3][2][0][:5])
```

```
1
[4, 5, ['tercera', 'valueError']]
['tercera', 'valueError']
valueError
E
ueError
terce
```

Agregar elemento a la lista

list.insert(indice,valor)

In [18]:
```
print(lespecial)
lespecial.insert(2,'me gusta paiton')
print(lespecial)
```

```
[1, 2, 3, [4, 5, ['tercera', 'valueError']]]
[1, 2, 'me gusta paiton', 3, [4, 5, ['tercera', 'valueError']]]
```

In [19]:
```
lespecial.append('append')
print(lespecial)
```

```
[1, 2, 'me gusta paiton', 3, [4, 5, ['tercera', 'valueError']], 'append']
```

In [5]:
```
frutas = ['manzana', 'platano', 'cereza', 'mango']
print(frutas)
print(len(frutas))
```

```
['manzana', 'platano', 'cereza', 'mango']
4
```

In [45]:
```
lmixta = ['ab', 45, True, 3.45, False, 34, 'x', True]
print(lmixta)
print(type(lmixta))
```

```
for elemento in lmixta:
    print(type(elemento), end="")   # end="" quita el salto de linea

    #TAREA: IMPRIMIR SOLO SI EL ELEMENTO ES BOOLEANO
print ("\n")

for elemento in lmixta:
    if type(elemento) == bool:
        print(elemento, " ", end="")
```

```
['ab', 45, True, 3.45, False, 34, 'x', True]
<class 'list'>
<class 'str'><class 'int'><class 'bool'><class 'float'><class 'bool'><class 'int'><c
lass 'str'><class 'bool'>

True  False  True
```

lista.remove()

elimina los elementos en una lista

In [6]:
```
print(frutas)
frutas.remove('cereza')
print(frutas)
```

```
['manzana', 'platano', 'cereza', 'mango']
['manzana', 'platano', 'mango']
```

In [7]:
```
print(frutas)
frutas.pop(2)
print(frutas)
```

```
['manzana', 'platano', 'mango']
['manzana', 'platano']
```

In [8]:
```
frutas.insert(1,'cereza')
frutas.append('cereza')
print(frutas)
```

```
['manzana', 'cereza', 'platano', 'cereza']
```

In [9]:
```
frutas.remove('cereza')
print(frutas)
```

```
['manzana', 'platano', 'cereza']
```

lista.clear()

limpia (elimina) el contenido de la lista

In [12]:
```
print(frutas)
frutas.clear()
print(frutas)
```

```
['manzana', 'platano', 'cereza']
[]
```

list.sort()

ordena los elementos de la lista ascendentes por default

In [ ]:
```
ltext=['naranja', 'mango', 'kiwi', 'piña','banana']
lnumero = [100,50,4,189,43]
print(ltext)
print(lnumero)

ltext.sort()

print(ltext)
print(lnumero)
```

```
['naranja', 'mango', 'kiwi', 'piña', 'banana']
[100, 50, 4, 189, 43]
['banana', 'kiwi', 'mango', 'naranja', 'piña']
[100, 50, 4, 189, 43]
```

lista.sort(reverse = true)

In [ ]:
```
print(ltext)
ltext.sort(reverse=True)
print(ltext)

cars = ['BMW','Volvo', 'Ford']
cars.sort(reverse = True)
print(cars)
```

```
['piña', 'naranja', 'mango', 'kiwi', 'banana']
['piña', 'naranja', 'mango', 'kiwi', 'banana']
['Volvo', 'Ford', 'BMW']
```

SETS (Conjuntos)

se usan para almacenar multipples elementosd de una sola variable

un conjunto es una coleccion que es ordenada, inmutable y sin indexar

no permite valores duplicados

Sin ordenar significa que los elemetos en un conjunto no tienen un orden definido

Los elementos de conjunto pueden aparecer en una orden diferente cada vez que se usa

y no puede ser referido por indice o clave

In [18]:
```
cfruta = {'manzana', 'fresa', 'kiwi'}
print(cfruta)
type(cfruta)
```

```
{'kiwi', 'fresa', 'manzana'}
```

Out[18]: set

In [2]:
```
cfruta = {'manzana', 'fresa', 'kiwi', 'manzana'}
print(cfruta)
```

```
{'manzana', 'fresa', 'kiwi'}
```

In [6]:
```
# conjunto mixto
# los valores True y 1 se consideran el  mismo valor en los conujntos
# y son tratados como duplicados

cmixto = {'apple', 'banana', 'cherry', True, 1,2,0, False}
print (cmixto)
```

```
{0, True, 2, 'banana', 'apple', 'cherry'}
```

In [ ]:
```
set1 = {'abc', 34, True,40,'hombre'}
print(set1)
```

---

tupla

una tupla es un coleccion que se ordenan, son inmutbales y permite valores duplicados

In [9]:
```
tnombre = ('Jose','gil','karen','jaqueline','elsa','roman','christian','yo','gil')
print(tnombre)
type(tnombre)
print(tnombre[5])
```

```
('Jose', 'gil', 'karen', 'jaqueline', 'elsa', 'roman', 'christian', 'yo', 'gil')
roman
```

Diccionarios

los diccionarios se utilizan para almacenar valores de datos en pares **clave-valor**

los elemenetos del diccionario:

- estan ordenados
- se pueden cambiar
- no permite duplicados

los elementos del diccionario se presdentan en pares clave-valor y se pueden hacer referencia a ellos mediante el nombre de la clave

In [15]:
```
dPp = {'edad': 18,
       'nombre': 'pepe',
       'apellido': 'perez',
       'estatura': 1.80
       }
print(dPp)
```

```
{'edad': 18, 'nombre': 'pepe', 'apellido': 'perez', 'estatura': 1.8}
```

In [13]:
```
print(dPp['nombre'])
print(dPp['estatura'])
```

```
pepe
1.8
```

In [1]:
```
dTespaña = {
        1: 'casillas',
        15: 'ramos',
        3: 'pique',
        11: 'capdevilla',
        5: 'puyol',
        14: 'xavi Alonso',
        6: 'iniesta',
        8: 'Javi Hernandez',
        7: 'Villa'}
print(dTespaña[8])
```

```
Javi Hernandez
```

In [11]:
```
for k, v in dTespaña.items():
    print(f'{k}\t->\t{v}')
```

```
1       ->      casillas
15      ->      ramos
3       ->      pique
11      ->      capdevilla
5       ->      puyol
14      ->      xavi Alonso
6       ->      iniesta
8       ->      Javi Hernandez
7       ->      Villa
```

In [12]:
```
numElem = len(dTespaña)
print(numElem)
print('numeros de elementos del diccionario = %i'%numElem)
print(f'numeros de elementos del diccionario = {numElem}')
```

```
9
numeros de elementos del diccionario = 9
numeros de elementos del diccionario = 9
```

In [13]:
```
eKeys = dTespaña.keys()
print(f'claves de diccionario= {eKeys}')
```

```
claves de diccionario= dict_keys([1, 15, 3, 11, 5, 14, 6, 8, 7])
```

In [8]:
```
eValues = dTespaña.values()
print(f'valores del diccionario dTespaña:\n{eValues}')
```

```
valores del diccionario dTespaña:
dict_values(['casillas', 'ramos', 'pique', 'capdevilla', 'puyol', 'xavi Alonso', 'in
iesta', 'Javi Hernandez', 'Villa'])
```

obtener elemento en un diccionario mediante su clave

In [15]:
```
num = 6
elem = dTespaña.get(num)
print(f'el elemento del diccionario con clave {num} es: {elem}')
```

```
el elemento del diccionario con clave 6 es: iniesta
```

---

funciones

def

una funcion es unv¿bloque de cpdigo que solo ejecuta cuando se llama.

puede pasar datos, conocidos como parametros a una funcion

como resultado, una funcion puede devoler datos

In [16]:
```
def hola():
    print('hola desde una funcion de python')

hola()
```

```
hola desde una funcion de python
```

In [ ]:
```
def s_arg(name):
    print(f'hola {name}')

s_arg('enrique')
```

```
hola enrique
```

In [19]:
```
def add(n1,n2):
    suma = n1 + n2
    print(f'{n1} + {n2} = {suma}')

add(5,6)
```

```
5 + 6 = 11
```

In [7]:
```
def pais(country="Mexico"):
    print(f"soy de {country}")

pais('rusia')
pais('brasil')
pais()
```

```
soy de rusia
soy de brasil
soy de Mexico
```

In [9]:
```python
# ARGUMENTOS POR PALABRA CLAVE
#   se puede enviar argumentos con la sintaxis clave = valor. Esto permite
#   que el odern de los argumentos  no importe .

def welcome (nombre,apellido):
    print(f'Binevenido a paiton {nombre} {apellido}')

#posicional
welcome('Pedro','Perez')

#nominal
welcome(apellido='Quino', nombre='Karla')
```

```
Binevenido a paiton Pedro Perez
Binevenido a paiton Karla Quino
```

RETURN

In [14]:
```python
def suma(num1,num2):
    sum = num1 + num2
    return sum
print(suma(5,6))
```

```
11
```

In [3]:
```python
# calcular el area de un triangulo

def area_triangulo (base,altura):
    area = base * altura/2
    return area
print (area_triangulo(4,5))
```

```
10.0
```

In [6]:
```python
# calcular el area de un triangulo

def area_triangulo (base,altura):
    return base * altura/2
b= int(input( "Escribe la base del triangulo: "))
a= int(input( "Escribe la altura del numero: "))
print (area_triangulo(b,a))
```

```
15.0
```

Valor = 40

Curso: Programación para Ciencia de Datos
Tarea: Practicas Unidad 1
Ver todos los envíos

Karla Guadalupe Quino Cinta
krlcinya23@gmail.com
Fecha de entrega: 4 de marzo ...

Cambiar usuario

2 de 8 Reiniciar preferencias de tabla

Página 1 de 14

Entrega

Enviado para calificar
Calificado
La tarea fue enviada 12 horas 31 mins antes de la fecha límite
Los estudiantes pueden editar este envío

Practicas_PCD.pdf 4 de marzo de 2025, 08:28

Comentarios (0)

Calificación

Calificación:

Practicas

Hoja de presentación	No contien todos los datos ***0 puntos***	Datos incompletos ***1 puntos***	Completo ***2 puntos***	era tu nombre
Indice	No contiene ***0 puntos***	Contiene ***2 puntos***		
Practicas	No contiene ***0 puntos***	Parcialmente ***15 puntos***	Todas ***25 puntos***	Extra clases ***30 puntos***
Conclusión	No contiene ***0 puntos***	Pequeña ***3 puntos***	Completa ***6 puntos***	

Calificación actual en el libro
40.00

Comentarios de retroalimentación

Notificar a estudiantes GUARDAR CAMBIOS GUARDAR Y MOSTRAR SIGUIENTE REINICIAR

INSTITUTO TECNOLOGICO SUPERIOR DE SAN ANDRÉS TUXTLA

GRUPO: 810 A

PROGRAMACION PARA CIENCIA DE DATOS

CATEDRATICO: ROGELIO TELONA TORRES

DOCUMENTACIÓN EXAMEN UNIDAD I

EQUIPO I

JOSE MANUEL ROSAS FAJARDO

JAQUELINE GATICA ANTELE

KARLA GUADALUPE QUINO CINTA

ROMAN OMAR FISCAL POLITO

INTRODUCCIÓN:

Dentro de esta documentación, desarrollamos un código con el lenguaje de Python para aplicar los conocimientos de la unidad 1, usando diccionarios, listas y condicionales. Este proyecto, parte de un sistema de inventario de productos, permite añadir o modificar productos, buscar productos por nombre, eliminar productos y listar todos los productos con sus precios. Con un enfoque práctico y autónomo, se creó una solución sencilla para manejar la información de manera eficiente y sentar las bases con los conocimientos adquiridos en la materia.

NOMBRE DEL PROYECTO

Sistema de inventario de producto

DESCRIPCION

El programa gestiona un inventario de productos con las funcionalidades de añadir o modificar productos, buscar productos por nombre, eliminar productos del inventario y listar todos los productos con sus precios.

ESTRUCTURA GENERAL

El siguiente código está compuesto por las siguientes partes principales:

Diccionario [inventario]: que almacena los productos y precios

Bucle while: el menú se mantiene hasta que el usuario elija la opción de salir

Condicionales: los cuales la elección del usuario ejecutara diferentes funcionalidades que se solicitan, añadir, buscar, borrar, listar.

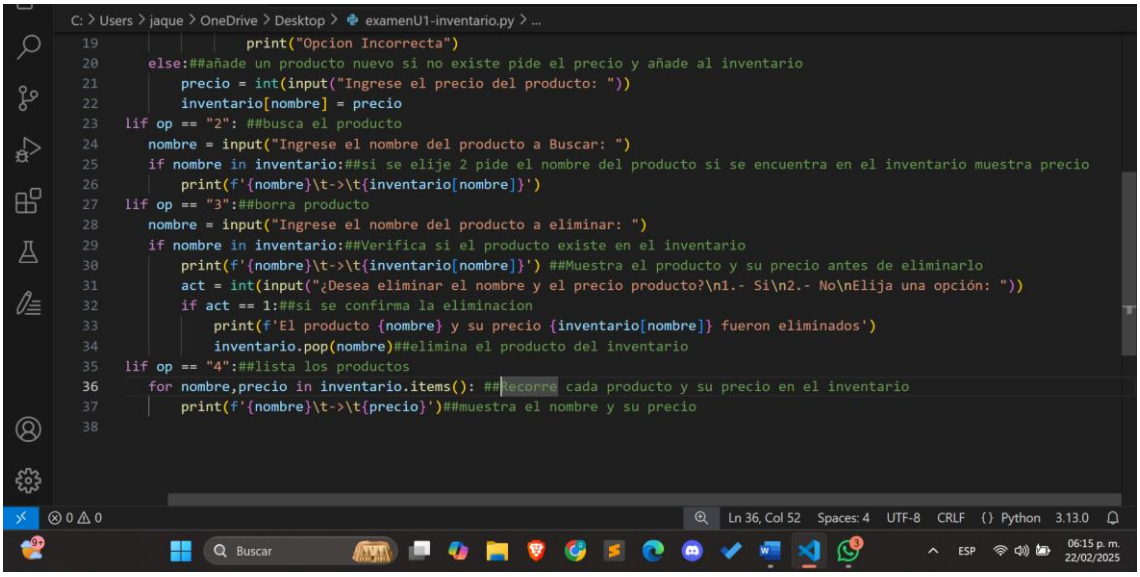
Examen Unidad 1

Proyecto: Sistema de inventario de producto

COMENTARIOS:

```python
inventario={} ## inicia un diccionario vacio con el nombre de inventario
```

```python
if op == "1": ##añadir o modificar producto
    nombre = input("Ingrese el nombre del producto a Añadir/Modificar: ")
    if nombre in inventario:##Si elige la opcion 1 el programa solicita el nombre del producto
        print(f'{nombre}\t->\t{inventario[nombre]}')##se verifica si el producto existe dentro del inventario
        act = 0 ##se actualiza el precio actual
        while act != 1 and act != 2:##se inicia un bucle interno donde el usuario elija una de las opciones
            act = int(input("¿Desea actualizar el precio del producto?\n1.- Si\n2.- No\nElija una opción: "))
            if act == 1:##si elije 1, solicita el nuevo precio y actualiza el diccionario
                precio = int(input("Ingrese el valor del producto: "))
                inventario[nombre] = precio
            elif act == 2:##si elije 2 cancela la actualizacion del diccionario
                print("Actualización cancelada")
            else:##de caso contrario imprime un mensaje
                print("Opcion Incorrecta")
    else:##añade un producto nuevo si no existe pide el precio y añade al inventario
        precio = int(input("Ingrese el precio del producto: "))
        inventario[nombre] = precio
```



ESTRUCTURA DEL CODIGO

Archivo/Módulo: El código se encuentra en un único archivo (por ejemplo, inventario.py) y se ejecuta de forma secuencial.

Inicialización: Se crean las variables y estructuras necesarias (un diccionario para el inventario y una variable para la opción del menú).

Bucle Principal: Se mantiene activo hasta que el usuario decide salir. Dentro de este bucle se muestra un menú interactivo y se ejecutan las operaciones correspondientes según la opción seleccionada.

Operaciones del Menú: Cada bloque if/elif dentro del bucle se encarga de una funcionalidad específica (añadir/modificar, buscar, borrar o listar productos).

AÑADIR/MODIFICAR PRODUCTO

Permitir al usuario que añada un producto nuevo al inventario o modificar el precio del producto existente.

Parámetros de Entrada (implícitos a través de input):

nombre: El nombre del producto a añadir o modificar.

precio: El valor numérico del producto, solicitado cuando se añade o se actualiza el precio.

act: La opción elegida por el usuario para confirmar si desea actualizar el precio de un producto ya existente (1 para sí, 2 para no).

Valor de Retorno: No devuelve ningún valor, pero actualiza el diccionario inventario modificando o añadiendo la clave nombre con su valor correspondiente (precio).

BUSCAR PRODUCTO

Propósito: Permitir al usuario buscar un producto en el inventario y, si existe, mostrar su precio.

Parámetro de Entrada:

nombre: El nombre del producto a buscar, obtenido mediante input.

Valor de Retorno: Imprime el nombre y el precio del producto si se encuentra en el inventario. No devuelve ningún valor.

BORRAR PRODUCTO

Propósito: Permitir al usuario eliminar un producto del inventario, previa confirmación.

Parámetro de Entrada:

nombre: El nombre del producto a eliminar, solicitado mediante input.

act: La opción elegida por el usuario para confirmar la eliminación (1 para sí, 2 para no).

Valor de Retorno: Elimina el producto del diccionario inventario si se confirma la acción. No devuelve ningún valor, pero imprime mensajes informativos.

LISTAR PRODUCTOS

Propósito: Mostrar al usuario todos los productos almacenados en el inventario junto a sus precios.

Parámetro de Entrada: No se requieren parámetros, ya que se recorre directamente el diccionario inventario.

Valor de Retorno: Imprime en pantalla cada producto con su precio. No devuelve ningún valor.

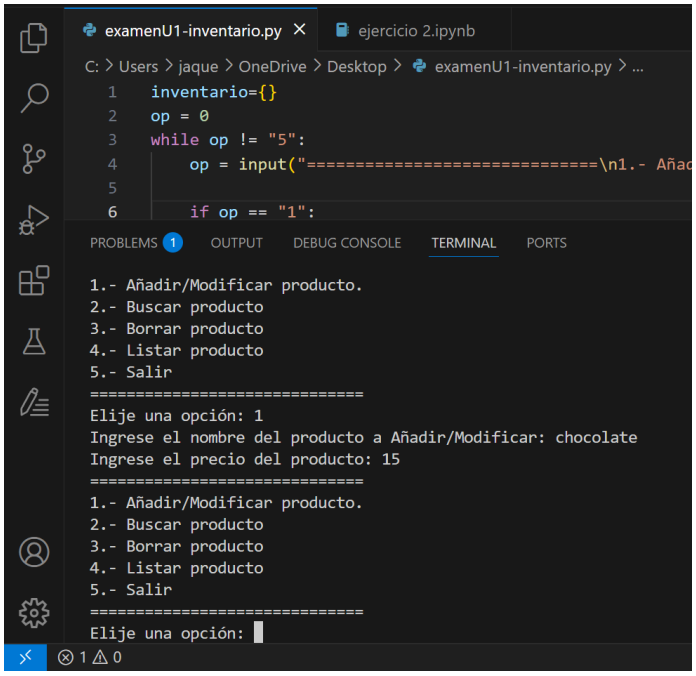
Examen Unidad 1

Proyecto: Sistema de inventario de producto

PRUEBAS Y RESULTADOS

1. Añadir/Modificar producto

```python
if op == "1": ##añadir o modificar producto
    nombre = input("Ingrese el nombre del producto a Añadir/Modificar: ")
    if nombre in inventario:##Si elige la opcion 1 el programa solicita el nombre del producto
        print(f'{nombre}\t->\t{inventario[nombre]}')##se verifica si el producto existe dentro del inventario
        act = 0 ##se actualiza el precio actual
        while act != 1 and act != 2:##se inicia un bucle interno donde el usuario elija una de las opciones
            act = int(input("¿Desea actualizar el precio del producto?\n1.- Si\n2.- No\nElija una opción: "))
            if act == 1:##si elije 1, solicita el nuevo precio y actualiza el diccionario
                precio = int(input("Ingrese el valor del producto: "))
                inventario[nombre] = precio
            elif act == 2:##si elije 2 cancela la actualizacion del diccionario
                print("Actualización cancelada")
            else:##de caso contrario imprime un mensaje
                print("Opcion Incorrecta")
    else:##añade un producto nuevo si no existe pide el precio y añade al inventario
        precio = int(input("Ingrese el precio del producto: "))
        inventario[nombre] = precio
```



2. Buscar producto

```
23      elif op == "2": ##busca el producto
24          nombre = input("Ingrese el nombre del producto a Buscar: ")
25          if nombre in inventario:##si se elije 2 pide el nombre del producto si se encuentra en el inventario muestra precio
26              print(f'{nombre}\t->\t{inventario[nombre]}')
```

```
==============================
1.- Añadir/Modificar producto.
2.- Buscar producto
3.- Borrar producto
4.- Listar producto
5.- Salir
==============================
Elije una opción: 2
Ingrese el nombre del producto a Buscar: chocolate
chocolate       ->      15
==============================
```

3. Borrar producto

```
27      elif op == "3":##borra producto
28          nombre = input("Ingrese el nombre del producto a eliminar: ")
29          if nombre in inventario:##Verifica si el producto existe en el inventario
30              print(f'{nombre}\t->\t{inventario[nombre]}') ##Muestra el producto y su precio antes de eliminarlo
31              act = int(input("¿Desea eliminar el nombre y el precio producto?\n1.- Si\n2.- No\nElija una opción: "))
32              if act == 1:##si se confirma la eliminacion
33                  print(f'El producto {nombre} y su precio {inventario[nombre]} fueron eliminados')
34                  inventario.pop(nombre)##elimina el producto del inventario
```

```
==============================
Elije una opción: 3
Ingrese el nombre del producto a eliminar: cacahuate
cacahuate       ->      20
¿Desea eliminar el nombre y el precio producto?
1.- Si
2.- No
Elija una opción: 1
El producto cacahuate y su precio 20 fueron eliminados
==============================
1.- Añadir/Modificar producto.
2.- Buscar producto
3.- Borrar producto
4.- Listar producto
5.- Salir
==============================
```

4. Listar productos

```
    elif op == "4":##lista los productos
        for nombre,precio in inventario.items(): ##Recorre cada producto y su precio en el inventario
            print(f'{nombre}\t->\t{precio}')##muestra el nombre y su precio

```

```
==============================
1.- Añadir/Modificar producto.
2.- Buscar producto
3.- Borrar producto
4.- Listar producto
5.- Salir
==============================
Elije una opción: 4
chocolate       ->      15
cacahuate       ->      20
paleta  ->      5
==============================
```

5. Salir

```
3   while op != "5":##Bucle principal que mantiene el programa en ejecución hasta que el usuario elija salir
```

```
==============================
1.- Añadir/Modificar producto.
2.- Buscar producto
3.- Borrar producto
4.- Listar producto
5.- Salir
==============================
Elije una opción: 5
PS C:\Users\jaque>
```

CONCLUSIÓN:

Este proyecto nos permitió crear un sistema de inventario usando **diccionarios**, lo que facilita almacenar productos con sus precios en pares clave-valor. Gracias a esto, pudimos **agregar, modificar, eliminar y buscar productos**, además de listar todo el inventario de forma organizada.

Los **condicionales (if, elif, else)** fueron esenciales para tomar decisiones dentro del programa, asegurando que cada acción se ejecutara correctamente según la opción del usuario. También utilizamos **bucles**, que nos permitieron recorrer el diccionario para

buscar o mostrar productos. Algo interesante fue el uso de for...else, que ejecuta el bloque else si el bucle termina sin interrupciones, lo que nos ayudó a manejar situaciones en las que un producto no existía.

En general, este ejercicio nos ayudó a entender mejor cómo se gestionan datos en Python y a aplicar estos conceptos a problemas reales como el control de inventario. Aunque aún hay mucho por mejorar, es un buen inicio para desarrollar programas más avanzados.

Valor = 40

Curso: Programación para Ciencia de Datos

Tarea: Examen Unidad 1 (Python)

Ver todos los envíos

Karla Guadalupe Quino Cinta

krlcinya23@gmail.com

Fecha de entrega: 24 de febrero ...

Cambiar usuario

2 de 7 Reiniciar preferencias de tabla

Página 1 de 13

Entrega

Enviado para calificar

Calificado

La tarea fue enviada 2 horas 56 mins antes de la fecha límite

Los estudiantes pueden editar este envío

documentacion_exam_U1__EQ1.pdf 24 de febrero de 2025, 20:03

Comentarios (0)

Calificación

Calificación:

Practicas

Funcionalidad del Código	El programa tiene errores graves o no cumple con los requisitos. ***0 puntos***	Algunas funciones presentan errores o no cumplen completamente con el objetivo. ***5 puntos***	Funciona correctamente con pequeños errores, pero cumple con los requisitos principales. ***8 puntos***	Todas las funciones del programa operan correctamente y sin errores. Se cumplen todos los requisitos del problema. ***10 puntos***	
Estructura y Organización del Código	Código desorganizado, difícil de entender, sin comentarios ni estructura clara. ***0 puntos***	Código funcional pero con mala organización y pocos comentarios. ***5 puntos***	Código mayormente organizado, aunque con algunas partes repetitivas o poco eficientes. ***8 puntos***	Código bien estructurado, modular y con nombres de variables y funciones adecuados. Uso correcto de comentarios. ***10 puntos***	
Validaciones y Manejo de Errores	No hay validaciones, lo que provoca errores en la ejecución. ***0 puntos***	Se implementan pocas validaciones; el programa puede fallar con datos inesperados. ***5 puntos***	Algunas validaciones implementadas, pero no cubren todos los casos. ***8 puntos***	Se implementan validaciones adecuadas para evitar errores del usuario. El programa maneja entradas incorrectas sin fallar. ***10 puntos***	
Pruebas y Corrección de Errores	No se realizaron pruebas ni se corrigieron errores antes de la entrega. ***0 puntos***	Se hicieron pocas pruebas y quedaron errores sin corregir. Informe incompleto. ***5 puntos***	Se realizaron pruebas básicas, aunque algunos errores persisten. Se incluye informe con correcciones. ***8 puntos***	Se realizaron pruebas exhaustivas y se corrigieron errores antes de la entrega. Se entrega un informe detallado de pruebas. ***10 puntos***	

Trabajo en Equipo y Cumplimiento de Roles	No hubo coordinación ni trabajo en equipo evidente. ***0 puntos***	Algunos miembros no participaron activamente o hubo problemas en la organización del equipo. ***5 puntos***	La mayoría de los miembros cumplieron su rol, aunque algunos participaron menos. Organización aceptable. ***8 puntos***	Todos los miembros participaron activamente y cumplieron su rol de manera equitativa. Buena comunicación y organización. ***10 puntos***	
Documentación y Manual de Usuario	Presentación deficiente o inexistente. No se logró demostrar la funcionalidad del código. ***0 puntos***	Presentación poco estructurada y con fallos en la demostración. Poca participación del equipo. ***5 puntos***	Presentación comprensible, aunque con pequeños problemas. Demostración funcional, pero con algunos errores. ***8 puntos***	Presentación bien organizada, con una explicación clara del código y una demostración funcional. Todos los miembros participaron. ***10 puntos***	
Presentación del Proyecto	Presentación deficiente o inexistente. No se logró demostrar la funcionalidad del código. ***0 puntos***	Presentación poco organizada y con errores en la demostración. Poca participación del equipo. ***5 puntos***	Presentación comprensible con una demostración funcional. Participación de la mayoría del equipo. ***8 puntos***	Presentación clara y bien estructurada. Demostración funcional del código. Todos los miembros participaron. ***10 puntos***	

Calificación actual en el libro
38.86

Comentarios de retroalimentación

Notificar a estudiantes

GUARDAR CAMBIOS GUARDAR Y MOSTRAR SIGUIENTE REINICIAR