

Bootcamp Inteligencia Artificial

Nivel Explorador

Semana 5: Python Para Inteligencia Artificial

Funciones, Listas y Tuplas, Diccionarios

Agenda

1. Funciones
2. Listas y Tuplas
3. Diccionarios

1.1 Funciones Integradas

➤ **Funciones integradas (built-in functions)**

Una función es un bloque de instrucciones agrupadas, que permiten reutilizar partes de un programa y Python incluye varias funciones de forma predeterminada.

Una función (en este contexto) es una parte separada del código de computadora el cual es capaz de:

- Efecto
- Resultado
- Argumento

1.2 Funciones Integradas

<code>abs()</code>	<code>dict()</code>	<code>help()</code>	<code>min()</code>	<code>setattr()</code>
<code>all()</code>	<code>dir()</code>	<code>hex()</code>	<code>next()</code>	<code>slice()</code>
<code>any()</code>	<code>divmod()</code>	<code>id()</code>	<code>object()</code>	<code>sorted()</code>
<code>ascii()</code>	<code>enumerate()</code>	<code>input()</code>	<code>oct()</code>	<code>staticmethod()</code>
<code>bin()</code>	<code>eval()</code>	<code>int()</code>	<code>open()</code>	<code>str()</code>
<code>bool()</code>	<code>exec()</code>	<code>isinstance()</code>	<code>ord()</code>	<code>sum()</code>
<code>bytearray()</code>	<code>filter()</code>	<code>issubclass()</code>	<code>pow()</code>	<code>super()</code>
<code>bytes()</code>	<code>float()</code>	<code>iter()</code>	<code>print()</code>	<code>tuple()</code>
<code>callable()</code>	<code>format()</code>	<code>len()</code>	<code>property()</code>	<code>type()</code>
<code>chr()</code>	<code>frozenset()</code>	<code>list()</code>	<code>range()</code>	<code>vars()</code>
<code>classmethod()</code>	<code>getattr()</code>	<code>locals()</code>	<code>repr()</code>	<code>zip()</code>
<code>compile()</code>	<code>globals()</code>	<code>map()</code>	<code>reversed()</code>	<code>__import__()</code>
<code>complex()</code>	<code>hasattr()</code>	<code>max()</code>	<code>round()</code>	
<code>delattr()</code>	<code>hash()</code>	<code>memoryview()</code>	<code>set()</code>	

1.3 Funciones Integradas

Dentro del listado anterior de funciones las más conocidas son:

- Función `print()`: imprime en pantalla el argumento.
- Función `input()`: Permite la entrada de datos al usuario

```
print("como te llamas compañero")
nombre=input()
print("hola:", nombre, "bienvenid@ a este curso de Python...")
```



```
Python 3.9.2 (tags/v3.9.2:1a79785, Feb 19 2021, 13:44:55) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:/Users/Liliana/Desktop/ejemplo función print e input.py =====
como te llamas compañero
Liliana Contreras
hola: Liliana Contreras bienvenid@ a este curso de Python...
>>> |
```

1.4 Funciones con parámetros de entrada

¿Qué sucede cuando Python encuentra una invocación como la que está a continuación?
nombreFunción(argumento)

En programación, así como en matemáticas, para las funciones definidas como:

$$f : A \rightarrow B,$$

- Al conjunto A se le denomina dominio y al conjunto B como rango.
- A partir de estos objetos se construye el encabezado de las funciones de programación.
- Sobre esta función se tiene que f corresponde al nombre de la función, el conjunto A corresponde al tipo de los argumentos de dicha función y el conjunto B que es el rango corresponderá al valor de retorno de dicha función.

1.5 Funciones para cadenas en Python

Función	Utilidad	Ejemplo	Resultado
len()	Determina la longitud en caracteres de una cadena.	len("Hola Python")	11
join()	Convierte en cadena utilizando una separación	Lista = ['Python', 'es'] -''.join(Lista)	'Python-es'
split()	Convierte una cadena con un separador en una lista	a = ("hola esto sera una lista") Lista2 = a.split() print (Lista2)	['hola', 'esto', 'sera', 'una', 'lista']
replace()	Reemplaza una cadena por otra	texto = "Manuel es mi amigo" print (texto.replace ('es', 'era'))	Manuel era mi amigo
upper()	Convierte una cadena en Mayúsculas	texto = "Manuel es mi amigo" texto.upper()	'MANUEL ES MI AMIGO'
lower()	Convierte una cadena en Minúsculas	texto = "MaNueL eS ml AmlgO" texto.lower()	'manuel es mi amigo'

1.6 Funciones varias en Python

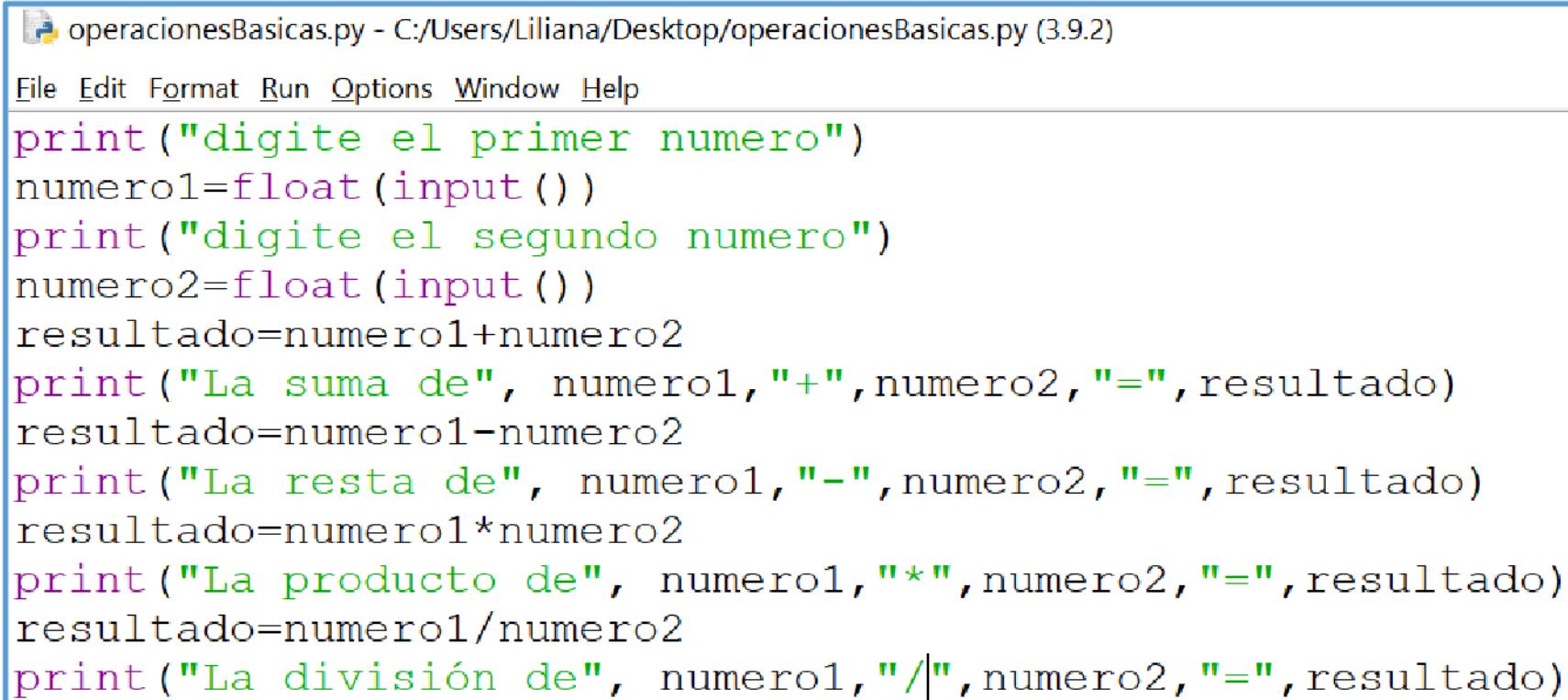
Función	Utilidad	Ejemplo	Resultado
range()	Crea un rango de números	x = range(5) print(list(x))	[0, 1, 2, 3, 4]
str()	Convierte un valor numérico a texto	str(22)	'22'
int()	Convierte a valor entero	int('22')	22
float()	Convierte un valor a decimal	float('2.22')	2.22
max()	Determina el máximo entre un grupo de números	x = [0, 1, 2] print(max(x))	2
min()	Determina el mínimo entre un grupo de números	x = [0, 1, 2] print(min(x))	0
sum()	Suma el total de una lista de números	x = [0, 1, 2] print(sum(x))	3

1.7 Funciones varias en Python

Función	Utilidad	Ejemplo	Resultado
list()	Crea una lista a partir de un elemento	x = range (+) print (list(x))	[0, 1, 2, 3, 4]
tuple()	Crea o convierte en una tupla	print(tuple(x))	(0, 1, 2, 3, 4)
open()	Abre, crea, edita un elemento (archivo)	with open("Ejercicios/Ejercicio.py", "w") as variables: variables.writelines("Eje")	Crea el archivo "Ejercicio.py" con el contenido "Eje"
ord()	Devuelve el valor ASCII de una cadena o carácter.	print(ord('A'))	65
round()	Redondea después de la coma de un decimal	print (round(12.723))	13
type()	Devuelve el tipo de un elemento	type(x)	<class 'range'>

1.8 Ejemplo

Realizar un programa que capture dos números enteros ingresados por el usuario y realice las operaciones aritméticas básicas (suma, resta, multiplicación y división)

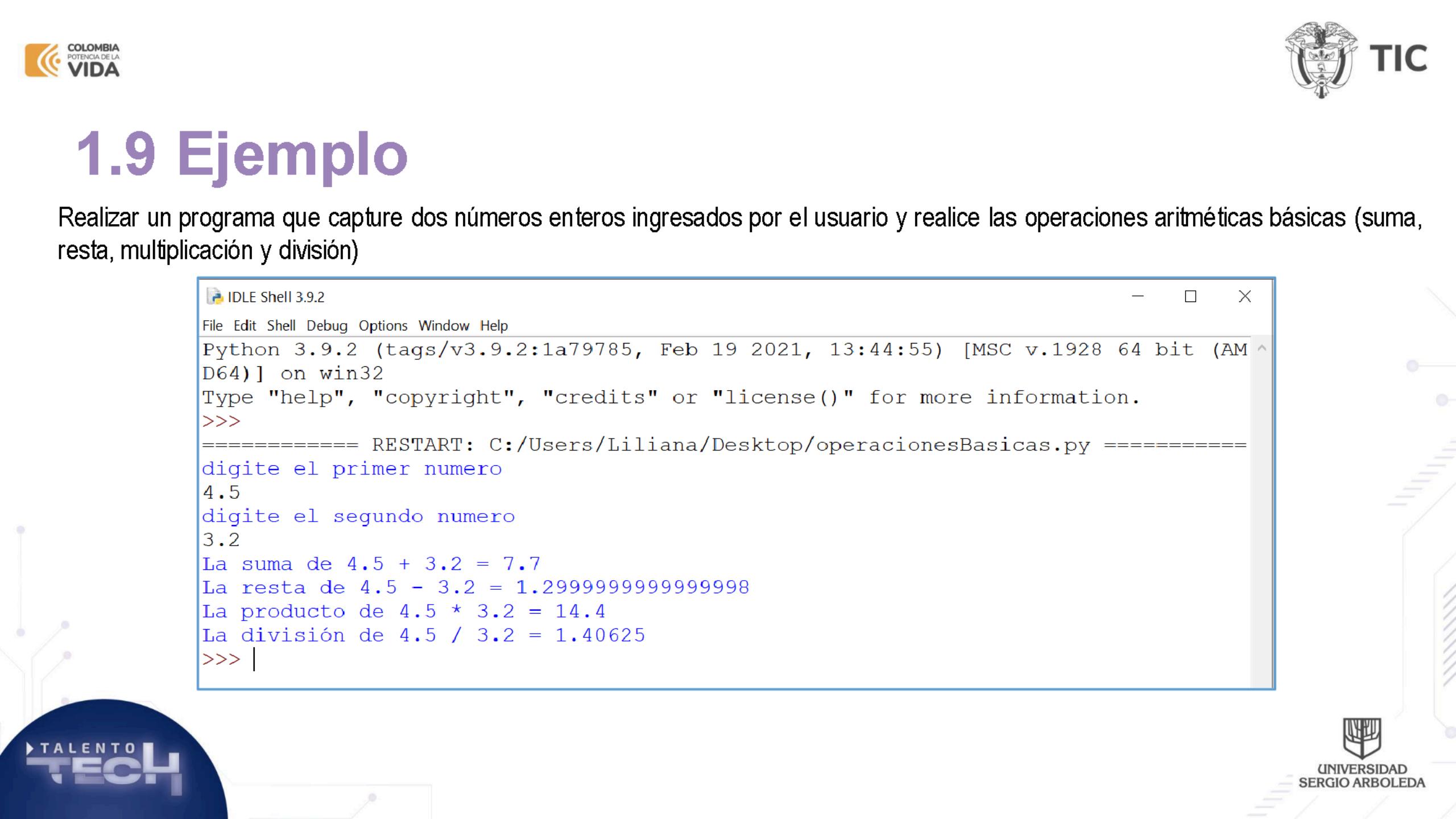


```
operacionesBasicas.py - C:/Users/Liliana/Desktop/operacionesBasicas.py (3.9.2)

File Edit Format Run Options Window Help
print("digite el primer numero")
numero1=float(input())
print("digite el segundo numero")
numero2=float(input())
resultado=numero1+numero2
print("La suma de", numero1,"+",numero2,"=",resultado)
resultado=numero1-numero2
print("La resta de", numero1,"-",numero2,"=",resultado)
resultado=numero1*numero2
print("La producto de", numero1,"*",numero2,"=",resultado)
resultado=numero1/numero2
print("La división de", numero1,"/",numero2,"=",resultado)
```

1.9 Ejemplo

Realizar un programa que capture dos números enteros ingresados por el usuario y realice las operaciones aritméticas básicas (suma, resta, multiplicación y división)



```
IDLE Shell 3.9.2
File Edit Shell Debug Options Window Help
Python 3.9.2 (tags/v3.9.2:1a79785, Feb 19 2021, 13:44:55) [MSC v.1928 64 bit (AM
D64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:/Users/Liliana/Desktop/operacionesBasicas.py =====
digite el primer numero
4.5
digite el segundo numero
3.2
La suma de 4.5 + 3.2 = 7.7
La resta de 4.5 - 3.2 = 1.2999999999999998
La producto de 4.5 * 3.2 = 14.4
La division de 4.5 / 3.2 = 1.40625
>>> |
```

1.10 Ejemplo

Realizar un programa que calcule la siguiente expresión:

((5 * ((25 mod 13) + 100) / (2 * 13)) // 2)

```
print( (5 * ((25 % 13) + 100) / (2 * 13)) // 2)
```

```
===== RESTART: C:/Users/Liliana/Desktop/operacionesBasicas.
10.0
>>> |
```

1.11 Ejemplo

Crea un programa que pida al usuario un número real y muestra su raíz cuadrada.

```
import math
print("Digite el número al que desea calcular la raiz cuadrada")
numero=float(input())
raizC=math.sqrt(numero)
print("La raiz cuadrada de ",numero, "es",raizC)
```

```
===== RESTART: C:\Users\Liliana\Desktop\operacionesBasicas.py =====
Digite el número al que desea calcular la raiz cuadrada
7.66
La raiz cuadrada de 7.66 es 2.7676705006196096
>>> |
```

1.12 Ejemplo

Crear un programa que permita solucionar un problema a través del teorema de Pitágoras:

```
print ("Ingrese el valor del cateto adyacente")
cA=float(input())
print ("Ingrese el valor del cateto opuesto")
cO=float(input())
Hipotenusa = (cA ** 2 + cO ** 2) ** 0.5
print("el valor de la Hipotenusa es=", Hipotenusa)
```

```
===== RESTART: C:\Users\Liliana\Desktop\operacionesBasicas.py =====
Ingrese el valor del cateto adyacente
5
Ingrese el valor del cateto opuesto
2
el valor de la Hipotenusa es= 5.385164807134504
>>> |
```

1.13 Ejercicios

1. Crea un programa que pida al usuario un número real y muestra su raíz cuarta (la raíz cuadrada de la raíz cuadrada).
2. Crear un programa que calcule y escriba el área para un circulo.
3. Elaborar un programa que calcule y escriba el precio de venta para un artículo, se tiene, el nombre del producto y el costo de producción; el precio de venta se calcula añadiendo el 120 % como utilidad y 15 % de impuestos.

2.1 Listas

- Una lista es una secuencia de elementos que puede almacenar datos heterogéneos tales como: enteros, reales, cadenas, tuplas, diccionarios y otros más, inclusive otras listas.
- Una lista se escribe como la secuencia de datos a mantener, separados por una coma (,), y delimitada por los paréntesis cuadrados (corchetes). A diferencia de las cadenas y las tuplas, las listas son mutables, esto es, se pueden modificar después de definidas.
 - Lista vacía []
 - Lista con un elemento ["un elemento"]
 - Lista de dos elementos [1,2]
 - Lista de varios elementos [1,2,3,"hola", 2.4, [1,2,3], "H"]
 - Lista asignada a una variable x=[]

2.2 Operadores y métodos Listas

Operadores

- Concatenar +
- Agregar al final (extend())
- Repetir (*)
- Comparar (<, <=, >, >=, ==, !=)
- Sub indice []

Métodos

- Longitud (len())
- Agregando elementos (append())
- Insertando elementos (insert())
- Eliminando elementos (remove())
- Sublista (slice ())
- Contando (Count())
- Buscando (index())
- Máximo y mínimo (max (), min())
- Ordenando (sort())
- Convertir a lista (list())
- Remover en una posición (pop())

2.3 Tuplas

- Una tupla es una secuencia de elementos que puede almacenar datos heterogéneos tales como: enteros, reales, cadenas, listas, diccionarios y otros más, inclusive otras tuplas.
- Una tupla se escribe como la secuencia de datos a mantener, separados por una coma (,), secuencia delimitada por los paréntesis redondos.
- Como las cadenas de caracteres, las tuplas son inmutables, esto es, no se pueden modificar después de definidas.

- Tupla vacía ()
- Tupla con un elemento (“un elemento”)
- Tupla con dos elementos (1,2)
- Tupla de varios elementos (1,2,”hola”, 2.9, (1,2,3), “mundo”)
- Se puede asignar una variable x= ()

2.4 Operadores y métodos Tuplas

Operadores

- Concatenar +
- Reperit *
- Comparar (<, <=, >, >=, ==, !=)
- Sub indice []

Métodos

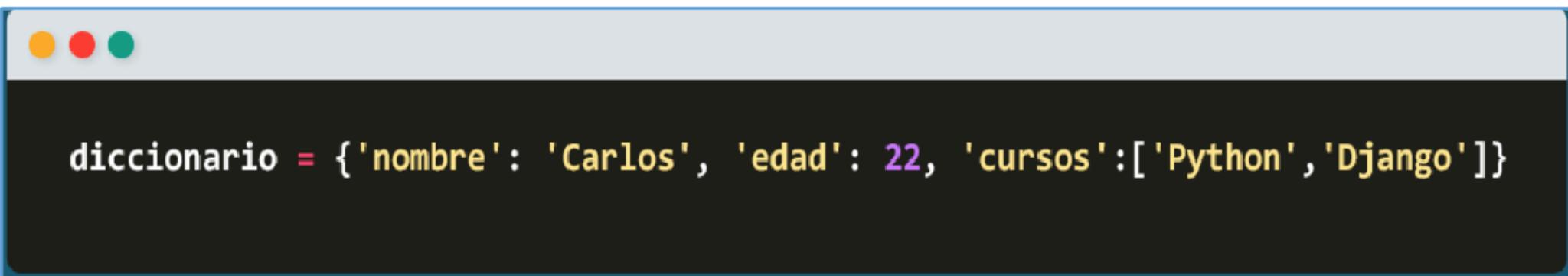
- Longitud (len())
- Sublista (slice ())
- Contando (Count())
- Buscando (index())
- Máximo y mínimo (max (), min())
- De cadena a tupla (tuple ())
- Desempacar variables (unpacking())
- map (map):
ejecuta una función para cada uno de los valores de la tupla

2.5 Ejercicios

1. Proponer una representación con tuplas para las cartas de una baraja.
2. Escribir una función poker que reciba cinco cartas de la baraja e informe si esas cartas forman o no un poker (es decir que hay 4 cartas con el mismo número).
3. Escribir una función que reciba dos vectores y devuelva su producto escalar.
4. Escribir una función que reciba dos vectores y devuelva si son o no ortogonales.
5. Escribir una función que reciba dos vectores y devuelva si son paralelos o no.
6. Escribir una función que reciba un vector y devuelva su norma.
7. Dada una lista de números enteros y un entero k , escribir una función que:
 - Devuelva tres listas, una con los menores, otra con los mayores y otra con los iguales a k .
 - Devuelva una lista con aquellos que son múltiplos de k .

3.1 Pareja Clave - Valor

- Una pareja clave-valor relaciona un par de objetos, la clave con qué se identifica o localiza a un objeto, y el valor que es el objeto referenciado.
- Las parejas clave-valor en Python se escriben en la forma clave:valor separando el objeto que va a servir de clave (a la izquierda) con un dos puntos del objeto valor, que será identificado por la clave.



```
diccionario = {'nombre': 'Carlos', 'edad': 22, 'cursos':[ 'Python', 'Django']}
```

3.2 Pareja Clave - Valor

➤ En Python se tienen estas restricciones:

- Una clave referencia a un sólo valor, pero un valor puede ser referenciado por diferentes claves.
- Cadena, enteros, reales y tuplas que no tenga listas como elementos, pueden ser usados como claves.
- La llave de un ítem no puede ser cambiada. Las llaves son únicas.
- Los valores pueden ser cualquier tipo de dato. Los valores pueden ser cambiados
- Un diccionario es una colección desordenada.

3.3 Diccionarios

Un diccionario es una colección de parejas clave-valor donde los valores pueden ser recuperados principalmente por su clave. Cada pareja clave-valor en un diccionario es considerada un ítem o registro. En Python, un diccionario se escribe separando los ítems por comas (,) y entre { }.

```
diccionario = {'nombre' : 'Carlos', 'edad' : 22, 'cursos': ['Python','Django','JavaScript'] }
```

```
print diccionario['nombre'] #Carlos
print diccionario['edad']#22
print diccionario['cursos'] #[ 'Python','Django','JavaScript']
```

3.4 Variables y Operadores

- ❖ Un diccionario se puede asignar una variable:

X={} Diccionario vacío

X={"A": "Un elemento"} Diccionario con una clave valor.

X={"A": "Un elemento", "B": "Dos elementos"} Diccionario con dos elementos.

Operadores

- ❖ Unir diccionarios (update()): agrega elementos de un diccionario a otro:
- ❖ Comparar (== !=)
- ❖ Accediendo []: accede al valor de un elemento con la clave dada
- ❖ Modificando []: se accede al valor y se modifica el elemento
- ❖ Eliminando (del ())

3.5 Métodos

- ✓ Longitud (len())
- ✓ Obteniendo valores (get())
- ✓ Máximo y mínimo (max(), min())
- ✓ Lista de claves y valores (keys, values)
- ✓ Convertir a diccionarios (dict())
- ✓ Eliminar entradas (clear())
- ✓ Copiar diccionarios (copy())

3.6 Ejercicios

- Escriba una función en python que pida un número entero positivo y que cree un diccionario cuyas claves sean desde el número 1 hasta el número indicado, y los valores sean los cuadrados de las claves.
- Escribe una función que reciba como parámetro una cadena y devuelva un diccionario con la cantidad de apariciones de cada carácter en la cadena.
- Crear un programa en python donde se va a declarar un diccionario para guardar los precios de distintas frutas. El programa pedirá el nombre de la fruta y la cantidad que se ha vendido y mostrará el precio final de la fruta a partir de los datos guardados en el diccionario. Si la fruta no existe nos dará un error. Tras cada consulta el programa nos preguntará si queremos hacer otra consulta.

Preguntas



