



Universidad Autónoma Chapingo

Departamento de Mecánica Agrícola
Ingeniería Mecatrónica Agrícola

Informe de practicas

Asignatura:

Visión por computadora

Nombre del profesor:

Luis Arturo Soriano Avendaño

Alumno:

Cocotle Lara Jym Emmanuel
[1710451-3]

GRADO:

6°

GRUPO:

7

Chapingo, Texcoco Edo. México

Fecha de entrega: 12/05/2021

Índice

Introducción	2
Desarrollo.....	3
Práctica 1.- Fundamentos de programación con Python	3
Objetivos.....	3
Código	3
Capturas	4
Práctica 2.- Manejo de variables	4
Objetivos.....	4
Código	5
Capturas	8
Práctica 3.- Manejo de Arrays.....	9
Objetivos.....	9
Código	9
Capturas	10
Práctica 4.- Manejo de funciones.....	10
Objetivos.....	10
Código	11
Capturas	14
Conclusión	14
Bibliografía.....	14

Introducción

La programación es el proceso por el cual a través de un lenguaje de programación podemos dar instrucciones, las cuales pueden ser ejecutadas por una computadora.

La dificultad que a menudo nos surge es el hecho que los problemas y sus soluciones son muy complejos. Estas estructuras y tipos de datos simples, suministrados por el lenguaje, si bien son ciertamente suficientes para representar soluciones complejas, están típicamente en desventaja a medida que trabajamos en el proceso de solución de

problemas. Requerimos maneras de controlar esta complejidad y contribuir con la creación de soluciones [1].

La visión por computadora busca modelar de manera matemática los procesos de percepción visual, y así mismo generar programas que permitan hacer uso de esta capacidad.

A través de las prácticas de este informe se busca comprender los conceptos básicos de la programación en Python, y así mismo hacer uso de estas funciones esenciales para programar.

Desarrollo

Práctica 1.- Fundamentos de programación con Python

Objetivos

- Instalación del programa de “Sublime Text 3”.
- Comprensión de los fundamentos de la programación en Python.
- Uso de librerías específicas para acciones específicas.
- Manejo de operadores.
- Manejo de tipo de números.

Código

```
1 import math
2
3 print('hola mundo') # Esto sirve para imprimir un texto
4 print(3*4) #Multiplicación
5 print((3*4)/((2**2)+(4/2))) #suma, resta, multiplicación, división y
potencia
6
7 #Utilizando una libreria
8 print(math.sqrt(4)) #Raiz cuadrada
9 print(math.sin(math.pi/2)) #La función seno evaluada en pi/2
10 print(math.exp(math.log(10))) #La función exponencial de logaritmo
natural
11 print(math.exp(math.log10(10))) #Función ex'ponencial de logaritmo base
10
12 print(math.exp(3/4)) #Funcion exponencialde una fracción
13 print(1/math.inf) #División de 1 entre infinito
14 print(math.inf*2) #Multiplicación de infinito por 2
15 print(math.inf/math.inf) #división de infinito sobre infinito
16
17 #Manejo de tipo de numeros
18 print(type(1234)) #Conocer el tipo de numero
19 print(type(3.2)) #Conocer el tipo de numero
```

```

20 print(type(2+5j)) #Conocer el tipo de numero
21 print(2+5j) #Generar un numero complejo opción 1
22 print(complex(2,5)) #Generar un numero complejo opción 2
23
24 #Manejo de notación científica
25 print(1e6) #Visualizamos una forma de expresar un millón
26 print(1e-3) #Visualizamos una forma de expresar una milésima
27
28 #manejo de operadores
29 print(5==4) #Preguntamos si los números son iguales
30 print(5>4) #Preguntamos si 5 es mayor que 4
31 print(5>=4) #Preguntamos si 5 es mayor o igual que 4
32 print(5<4) #Preguntamos si 5 es menor que 4
33 print(5<=4) #Preguntamos si 5 es menor o igual que 4
34 print(5!=4) #Preguntamos si 5 es diferente de 4
35
36 print((1+3)>(2+5)) #Preguntamos si 1+3 es mayor que 2+5
37 print((3>2)+(5>4)) #Preguntamos si 3 es mayor que 2 y si 5 es mayor que
4
38 print((14*24+60+60)>1000000) #Preguntamos si 14*24+60+60 es mayor que
1000000

```

Capturas

The screenshot shows a Sublime Text editor window with a Python file named 'Practica1.py'. The code in the file is as follows:

```

1 import math
2
3 print('hola mundo') # Esto sirve para imprimir un texto
4 print(3*4) #Multiplicación
5 print((3*4)/((2**2)*(4/2))) #suma, resta, multiplicación, división y potencia
6
7 #Utilizando una libreria
8 print(math.sqrt(4)) #Raiz cuadrada
9 print(math.sin(math.pi/2)) #La función seno evaluada en pi/2
10 print(math.exp(math.log(10))) #La función exponencial de logaritmo natural

```

The output of the script is displayed in the console below the editor:

```

hola mundo
12
2.0
2.0
1.0
10.000000000000002
2.718281828459045
2.117000016612675
0.0
inf
nan
<class 'int'>
<class 'float'>
<class 'complex'>
(2+5j)
1000000.0
0.001
False
True
True
False
False
True
False
2
False
ffinished in 1.2s

```

1.- Resultados de la práctica 1.

Práctica 2.- Manejo de variables

Objetivos

- Manejo de variables para operaciones u otros usos.

- Uso de cadenas para crear estructura de tipo lista, concatenar listas, estructuras de tipo tupla y de tipo diccionario.
- Conocer las características de las diferentes estructuras.

Código

```

1 #Manejo de variables
2
3 x = 1 #Declaracion de una variable
4 print(x) #impresión de una variable
5
6 y=2 #Declaramos otra variable
7 print(y*3) #Multiplicacion de una variable por un escalar
8
9 x = x + 1 #Reasignamos el valor de la variable
10 print(x) # Visualizamos el resultado
11
12 mensaje1= "Hola" #Declaramos una variable de tipo cadena
13 mensaje2= "mundo" #Declaramos una variable de tipo cadena
14
15 print(mensaje1+mensaje2) #Concatenación de dos variables de tipo cadena
16 print(mensaje1+" "+mensaje2) #Concatenación de variables de tipo cadena
17
18 mensaje_completo = mensaje1+" "+mensaje2 #Definimos una variable de
tipo cadena
19 print(len(mensaje_completo)) #Contamos los caracteres de la variable
20
21 print(mensaje_completo[9]) #Acceder a cualquier espacio dentro del
mensaje
22 print(mensaje_completo[2:7]) #Accedemos a un fragmento de la cadena
23 print(mensaje_completo[5:]) #Accedemos a un espacio completo de la
variable desde un punto
24 print(mensaje_completo[:4]) #Accedemos a un espacio completo de la
variable desde un punto
25
26 print("El valor de y="+str(y)) #Convertimos y concatenamos el valor de
una variable int
27
28 lema = "Solo aquel que esté dispuesto a morir debería tener el poder
de matar!!" #Declaramos una variable de tipo cadena
29
30 print(lema) #Visualizamos el contenido de la variable
31 print(lema.upper()) #Cambiamos de minúsculas a mayúsculas todas las
letras del mensaje
32 print(lema.count("e")) #Contamos todas las letras e

```

```

33 print(lema.count("a")) #Contamos todas las letras a
34 print(lema.replace("morir","vivir")) #Reemplazamos la palabra morir por
vivir en el lema
35
36
37 universidad = "Chapingo" #Declaramos una variable de tipo cadena
38 departamento = "DIMA" #Declaramos una variable de tipo cadena
39
40     print("El     mejor     departamento     de     %s     es
el %s"%(universidad,departamento)) #Imprimimos el mensaje con las variables
declaradas
41 print(f"El mejor departamento de {universidad} es el {departamento}")
#Imprimimos el mensaje con las variables declaradas de manera diferente
42
43 #Estructura de tipo lista
44 lista_numeros = [1,2,3] #Declaramos una variable de tipo lista
45 print(lista_numeros) #Visualizamos el contenido de la variable
46
47 lista_palabras = ["Manzana","Pera","Guayaba"] #Declaramos una variable
de tipo lista
48 print(lista_palabras) #Visualizamos el contenido de la variable
49
50 lista_concatenada = [1,2,3,"Manzana","Pera","Guayaba"] #Declaramos la
lista concatenada
51 print(lista_concatenada) #Visualizamos el contenido de la variable
52
53 lista_concatenada2 = [lista_números,lista_palabras] #Concatenamos 2
listas
54 print(lista_concatenada2) #Visualizamos el contenido de la variable
55
56 print(lista_concatenada[0]) #Accesamos al valor 0 de la lista
57 print(lista_concatenada[3]) #Accesamos al valor 3 de la lista
58 print(lista_concatenada[5]) #Accesamos al valor 5 de la lista
59
60 lista_concatenada3 = lista_numeros+lista_palabras #Concatenamos dos
listas
61 print(lista_concatenada3) #Visualizamos el contenido de la variable
62
63 lista_concatenada3.insert(3,"sandia") #Insertamos un nuevo valor a la
lista
64 print(lista_concatenada3) #Visualizamos el contenido de la variable
65
66 del lista_concatenada3[6] #Borramos el valor ubicado en el espacio 6
67 print(lista_concatenada3) #Visualizamos el contenido de la variable
68
69 #Estructura de tuplas

```

```

70 variable_tupla = (1,2,3,4) #Declaramos una variable de tipo tupla
71 print(variable_tupla) #Visualizamos el contenido de la variable tupla
72 print(len(variable_tupla)) #Contamos los elementos de la variable tupla
73 print(variable_tupla[1:3]) #Accesamos al intervalo de valores dentro
de la variable
74
75 lista_tuplas = [("Manzana",1),("Naranja",2),("Pera",3)] #Declaramos
una variable de tipo tupla
76 print(lista_tuplas) #Visualizamos el contenido de la variable tupla
77
78 a,b,c = lista_tuplas #Declaramos una variable de tipo tupla
79 print(a) #Visualizamos el contenido de la variable tupla
80 print(b) #Visualizamos el contenido de la variable tupla
81 print(c) #Visualizamos el contenido de la variable tupla
82
83 #Estructura set
84 datos = {3,3,2,1,4,5,6,4,2} #Declaramos una variable tipo set
85 print(datos) #Visualizamos el contenido de la variable set
86
87 lista_enteros = [1,2,3,3,2,1,2] #Declaramos una variable de tipo lista
88 print(set(lista_enteros)) #Visualizamos el contenido de la lista
89
90 fruta = "Manzana" #Declaramos una variable de tipo cadena
91 print(set(fruta)) #Visualizamos el contenido de la variable
92
93 #Estructura de tipo diccionario
94 diccionario = {"Apple":3,"Orange":6,"Watermelon":1} #Declaramos una
variable de tipo diccionario
95 print(diccionario) #Visualizamos su valor
96
97 print(diccionario["Apple"]) #Accesamos al valor del diccionario con la
palabra apple
98 print(diccionario.keys()) #Conocemos todas las palabras clave del
diccionario
99 print(diccionario.values()) #Conocemos todas las definiciones del
diccionario
100 print(len(diccionario)) #Contamos los elementos del diccionario
101
102
diccionario_string =
{"Apple":"tres","Orange":"seis","Watermelon":"uno"} #Declaramos una
variable de tipo diccionario con valores tipo cadena
103 print(diccionario_string["Apple"]) #Accesamos al valor del diccionario
con la palabra apple
104 print(diccionario_string.keys()) #Conocemos todas las palabras clave
del diccionario

```

```

105 print(diccionario_string.values()) #Conocemos todas las definiciones
del diccionario
106 print(len(diccionario_string)) #Contamos los elementos del diccionario
107
108 print("Apple" in diccionario) #Buscamos esta palabra clave "Apple" en
el diccionario
109 print("Sandia" in diccionario) #Buscamos esta palabra clave "Sandia"
en el diccionario
110 print(list(diccionario)) #Visualizamos las palabras del diccionario sin
sus valores

```

Capturas

```

C:\Users\jymc\Documents\Chapingo\6 Semestre\Visión por computadores\Programas\Practica2.py (Programas) - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help

HOLDERS
Programas
  > _pycache_
  Practica1.py
  Practica2.py
  Practica3.py
  Practica4.py
  Practica5.py

1 #Manejo de variables
2
3 x = 1 #Declaracion de una variable
4 print(x) #Impresión de una variable
5
6 y=2 #Declaramos otra variable
7 print(y*3) #Multiplicacion de una variable por un escalar.
8
9 x = x + 1 #Reasignamos el valor de la variable
10
1
6
2
Holamundo
Hola mundo
10
0
la mu
mundo
Hola
El valor de y=2
Solo aquel que este dispuesto a morir debería tener el poder de matar!!
SOLO AQUEL QUE ESTE DISPUESTO A MORIR DEBERIA TENER EL PODER DE MATAR!!
12
5
Solo aquel que este dispuesto a vivir debería tener el poder de matar!!
El mejor departamento de Chapingo es el DIMA
El mejor departamento de Chapingo es el DIMA
[1, 2, 3]
['Manzana', 'Pera', 'Guayaba']
[1, 2, 3, 'Manzana', 'Pera', 'Guayaba']
[[1, 2, 3], ['Manzana', 'Pera', 'Guayaba']]
1
Manzana
Guayaba
[1, 2, 3, 'Manzana', 'Pera', 'Guayaba']
[1, 2, 3, 'Sandia', 'Manzana', 'Pera', 'Guayaba']
{1: 2, 3: 'Sandia', 'Manzana': 'Pera', 1}

```

2.- Resultados de la práctica 2.

The screenshot shows a Sublime Text editor window with a file named 'Practica2.py'. The code in the editor is as follows:

```
1 #Manejo de variables
2
3 x = 1 #Declaracion de una variable
4 print(x) #Impresión de una variable
5
6 y = 2 #Declaramos otra variable
7 print(y*3) #Multiplicacion de una variable por un escalar.
8
9 x = x + 1 #Reasignamos el valor de la variable
```

The output of the script is displayed in the console at the bottom of the editor:

```
Manzana
Guayaba
[1, 2, 3, 'Manzana', 'Pera', 'Guayaba']
[1, 2, 3, 'sandia', 'Manzana', 'Pera', 'Guayaba']
[1, 2, 3, 'sandia', 'Manzana', 'Pera']
(1, 2, 3, 4)
4
(2, 3)
[('Manzana', 1), ('Naranja', 2), ('Pera', 3)]
('Manzana', 1)
('Naranja', 2)
('Pera', 3)
{1, 2, 3, 4, 5, 6}
{1, 2, 3}
{'a', 'M', 'z', 'n'}
{'Apple': 3, 'Orange': 6, 'Watermelon': 1}
3
dict_keys(['Apple', 'Orange', 'Watermelon'])
dict_values([3, 6, 1])
3
tres
dict_keys(['Apple', 'Orange', 'Watermelon'])
dict_values(['tres', 'seis', 'uno'])
3
True
False
['Apple', 'Orange', 'Watermelon']
[finished in 0.3s]
```

3.- Resultados de la práctica 2.

Práctica 3.- Manejo de Arrays

Objetivos

- Uso de la librería “NumPy” para la creación de matrices.
- Comprender algunas funciones que posee la librería “NumPy”.

Código

1 # Agregamos la librería numpy con el comando: "pip install numpy" con el símbolo de sistema

2

3 # Estructura Array

4 **import numpy as np** # Importamos la librería

5

6 **x = np.array([1,4,3])** #Construimos un vector

7 **print(x)** # Imprimimos el vector tipo fila

8

9 **y = np.array([[1,4,3],[9,2,7]])** # Construimos una matriz de 2 filas 3 columnas

10 **print(y)** # Imprimimos la matriz

11

12 **print(y.size)** # Conocemos el número de elementos de la matriz

13 **print(y.shape)** # Conocemos el número de filas y columnas de la matriz

14

15 **z = np.arange(0,11,1)** # construir un arreglo de datos desde 0 hasta 10 de uno en uno

16 **print(z)** # Visualizamos el arreglo

17

```

18 w = np.arange(-5,2,0.5) # Construir un arreglo de datos desde -5 hasta
2 de 0.5 en 0.5
19 print(w) # Visualizamos el arreglo
20
21 print(y[0,1]) # Accedemos al valor de la matriz en la fila 0 y columna 1
22 print(y[0,:]) # Accedemos a todos los valores de la fila 0
23 print(y[:,-1]) # Accedemos a los valores de la última columna
24
25 matriz_ceros = np.zeros((2,2)) # Construimos una matriz de 2x2 de ceros
26 matriz_unos = np.ones((3,3)) # Construimos una matriz de 3x3 de unos
27
28 print(matriz_ceros) # Visualizamos a la matriz de 2x2
29 print(matriz_unos) # Visualizamos a la matriz de 3x3

```

Capturas

The screenshot shows a Sublime Text editor window with a file named 'Practica3.py'. The code in the editor is as follows:

```

1 # Agregamos la libreria numpy con el comando: "pip install numpy" con el simbolo de sistema
2
3 # Estructura Array
4 import numpy as np # Importamos la libreria numpy
5
6 x = np.array([1,4,3]) #Construimos un vector
7 print(x) # Imprimos el vector tipo fila
8
9 y = np.array([[1,4,3],[9,2,7]]) # Construimos una matriz de 2 filas 3 columnas
10 print(y) # Imprimos la matriz

```

The output of the script is displayed in the console at the bottom of the editor:

```

[1 4 3]
[[1 4 3]
 [9 2 7]]
6
(2, 3)
[ 0  1  2  3  4  5  6  7  8  9 10]
[-5. -4.5 -4. -3.5 -3. -2.5 -2. -1.5 -1. -0.5  0.  0.5  1.  1.5]
4
[1 4 3]
[3 7]
[[0. 0.]
 [0. 0.]]
[[1. 1. 1.]
 [1. 1. 1.]
 [1. 1. 1.]]
[Finished in 0.4s]

```

4.- Resultados de la práctica 3.

Práctica 4.- Manejo de funciones

Objetivos

- Conocer los 4 tipos de funciones que existen, así como su funcionamiento.
- Identificar las partes que componen a una función.
- Uso de librerías en la creación de funciones.
- Identificar las diferencias entre variable local y variable global.
- Conocer la estructura y funcionamiento de una función anidada.
- Conocer el funcionamiento de una función lambda.

Código

```
1 # Manejo de funciones
2 # Algunos ejemplos son los que ya hemos trabajado en la librería math
3 # sin, cos, len, array
4
5 import numpy as np # Importamos la librería numpy
6
7 # Estructura para definir una función
8 def nombre_funcion(parametro1, parametro2): #Creamos una función que
recibe 2 parámetros
9     print(2+2) #Imprimimos la operación de 2+2
10
11 # Existen 4 tipos de funciones
12
13 # a) la función que recibe y devuelve valores
14 def suma_digitos(a,b): #Creamos una función que recibe dos valores
15     operacion_salida = a+b #Creamos una operación en la que sumamos
los dos valores recibidos
16     return operacion_salida #Devolvemos el valor de la operación
17
18 # b) La función que recibe valores, pero no devuelve valores
19 def suma_digitos2(a,b): #Creamos una función que recibe dos valores
20     suma_parametros = a+b #Creamos una operación en la que sumamos los
dos valores recibidos
21     print(suma_parametros) #Visualizamos la operación
22
23 # c) La función que no recibe valores y no devuelve valores
24 def suma_digitos3(): #Creamos una función la cual no recibe valores
25     print(3+4) #Visualizamos la operación de 3+4
26
27 # d) La función que no recibe valores, pero si devuelve valores
28 def suma_digitos4(): #Creamos una función la cual no recibe valores
29     suma_parametros = 3+4 #Creamos una operación en la que sumamos dos
valores establecidos
30     return(suma_parametros) #Devolvemos el valor de la operación
31
32 # Llamado de funciones
33 print(suma_digitos(3,4)) # Llamamos a la función tipo a)
34 suma_digitos2(5,9) # Llamamos a la función tipo b)
35 suma_digitos3() # Llamamos a la función tipo c)
36 print(suma_digitos4()) # Llamamos a la función tipo d)
37
38 # Regresar más de un valor en una función
39 def suma_trigonometrica(a,b): #Creamos una función que recibe dos
valores
```

```

40     salida1 = np.sin(a) + np.cos(b) #Con ayuda de la librería numpy
creamos una operación en la que sumamos el seno con el coseno de los
valores recibidos
41     salida2 = np.sin(b) + np.cos(b) #Con ayuda de la librería numpy
creamos una operación en la que sumamos el seno con el coseno de los
valores recibidos
42     return salida1,salida2,[salida1,salida2] #Devolvemos los valores
obtenidos de las operaciones
43
44     print(suma_trigonometrica(2,3)) #Hacemos uso de la función creada con
los valores de 2 y 3
45
46     def imprimir_porra(frase1="Jauri",frase2="ra"): #Creamos una función
con dos cadenas establecidas
47         print(f"{frase1} ju, {frase1} ja, Chapingo {frase2}, toros
salvajes {frase2}{frase2}{frase2}") #Visualizamos un texto mandando a
llamar las cadenas establecidas
48
49     imprimir_porra() #Mandamos a llamar a la función creada
50
51     #Ejemplos de variables locales y variables globales
52     #Ejemplo de variable local
53     def suma_variables(a,b,c): #Creamos una función que recibe 3 valores
54         operacion = a+b+c #Sumamos los valores recibidos
55         print(f"El valor de la variable adentro de la funcion es
{operacion}") #Visualizamos el resultado de la suma
56         return operacion #Devolvemos el valor obtenido de la operación
57
58     operacion = 1 #Igualamos a la variable operación con 1
59     suma_variables(1,2,3) #Mandamos a llamar a la función creada ay a las
variables les damos el valor de 1, 2, 3 respectivamente
60
61     print(f"El valor de la variable afuera de la funcion es {operacion}")
#Visualizamos el valor de la variable que colocamos fuera de la función
62
63     #Ejemplo de variable global
64     n = 24 #Declaramos una variable
65     def funcion(): #Creamos una función
66         global n #Se manda a llamar a la variable global n
67         print(f"Adentro de la funcion el valor de n es {n}") #Visualizamos
el valor de la variable global
68         n = 3 #Reasiganmos el valor de la variable global
69         print(f"Adentro de la funcion el nuevo valor asignado de n es
{n}") # Visualizamos el nuevo valor de la variable
70
71     funcion() #Llamamos a la función creada

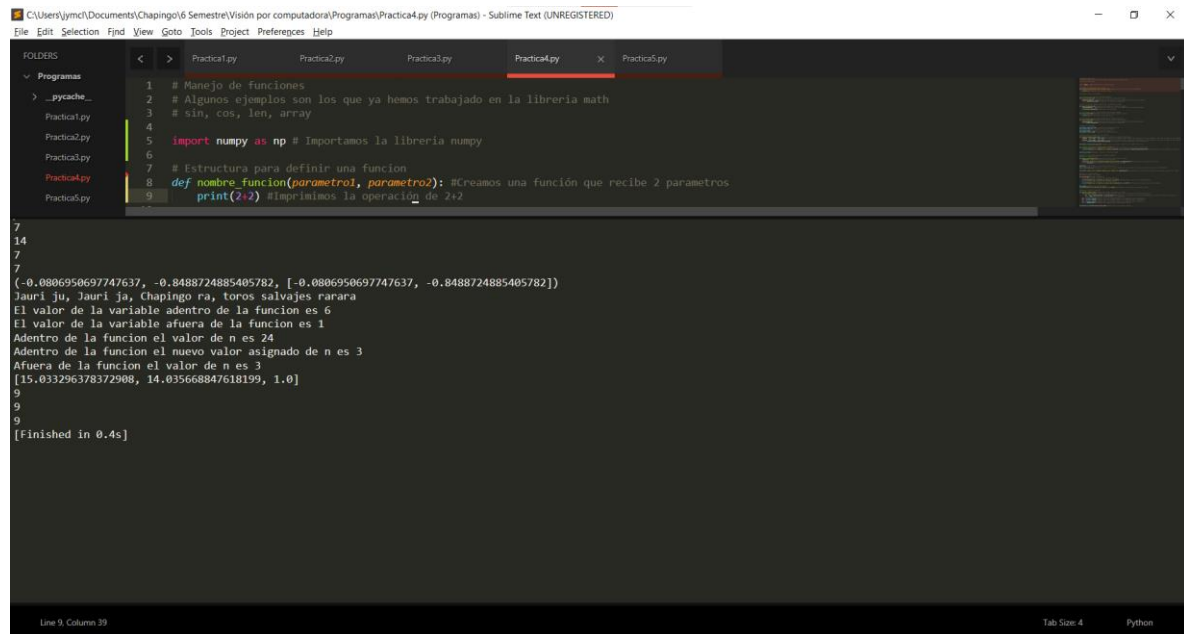
```

```

72 print(f"Afuera de la funcion el valor de n es {n}") #Visualizamos el
valor de la variable n actual
73
74 #Ejemplo de funciones anidadas
75 def funcion_xyz(x,y,z): #Creamos una función que recibe 3 valores
76     def funcion_xy(x,y): #Creamos una segunda función dentro de la
función la cual recibe dos valores
77         out = np.sqrt((x[0]-y[0])**2+(x[1]-y[1])**2) #Realizamos
una operación en la cual referenciamos los valores en base a la posición
78         return out #Devolvemos el valor obtenido de la operación
79     d0 = funcion_xy(x,y) #Hacemos uso de la segunda función y la
igualamos con la variable d0
80     d1 = funcion_xy(x,z) #Hacemos uso de la segunda función y la
igualamos con la variable d1
81     d2 = funcion_xy(y,z) #Hacemos uso de la segunda función y la
igualamos con la variable d2
82     return [d0,d1,d2] #Devolvemos los valores de las variables d0, d1,
d2
83
84 print(funcion_xyz((15,0),(0,1),(1,1))) #Hacemos uso de la función
anidada
85
86 #Función lambda
87
88 print(3**2) #Visualizamos el cuadrado de 3
89
90 cuadrado = lambda x:x**2 #Creamos una función lambda con la cual
obtenemos el cuadrado de un numero
91
92 print(cuadrado(3)) #Visualizamos el valor del cuadrado de 3 usando la
función lambda
93
94 def variable_cuadrado(x): #Creamos una función con una variable de
entrada
95     salida = x**2 #Realizamos una operación en la que obtenemos el
cuadrado de un número y lo igualamos con una variable
96     return salida #Devolvemos la variable de salida
97 print(variable_cuadrado(3)) #Hacemos uso de la función visualizando el
cuadrado de 3

```

Capturas



The screenshot shows a Sublime Text editor window with the file path `C:\Users\jmc\Documents\Chapingo\6 Semestre\Visión por computadora\Programas\Practica4.py (Programas) - Sublime Text (UNREGISTERED)`. The editor has five tabs open, with `Practica4.py` selected. The code in the editor is as follows:

```
1 # Manejo de funciones
2 # Algunos ejemplos son los que ya hemos trabajado en la libreria math
3 # sin, cos, len, array
4
5 import numpy as np # Importamos la libreria numpy
6
7 # Estructura para definir una funcion
8 def nombre_funcion(parametro1, parametro2): #Creamos una función que recibe 2 parametros
9     print(2+2) #Imprimimos la operación de 2+2
```

The output console at the bottom shows the following text:

```
7
14
7
7
(-0.0806950697747637, -0.8488724885405782, [-0.0806950697747637, -0.8488724885405782])
Jauri ju, Jauri ja, Chapingo ra, toros salvajes rarara
El valor de la variable adentro de la funcion es 6
El valor de la variable afuera de la funcion es 1
Adentro de la funcion el valor de n es 24
Adentro de la funcion el nuevo valor asignado de n es 3
Afuera de la funcion el valor de n es 3
[15.033296378372908, 14.035668847618199, 1.0]
9
9
9
[Finished in 0.4s]
```

5.- Resultados de la práctica 4.

Conclusión

A través de estas 4 practicas pude comprender y entender los fundamentos básicos de la programación en Python, además de poder hacer uso de estos para la creación de programas que, aunque son básicos, son indispensables para poder entender a detalle como lo ejecuta el compilador. El entendimiento claro de estos fundamentos son indispensables ya que es a partir de estos que se pueden resolver problemas más complejos en la programación.

Bibliografía

1. 1.4. ¿Qué es programación? — Solución de problemas con algoritmos y estructuras de datos. (2021). Consultado el 11 de Mayo del 2021, de <https://runestone.academy/runestone/static/pythoned/Introduction/QueEsProgramacion.html>