UNIVERSIDAD NACIONAL DEL ALTIPLANO

FACULTAD DE INGENIERIA ESTADISTICA E INFORMATICA

Escuela Profesional de Estadística e Informática

Actividad 2:

Definiciones : Variables, funciones ycRestricciones

Ingeniero: Fred Torres Cruz
Curso: Metodos de Optimizacion
Estudiante: Marco Paul Mamani Rodriguez
Codigo: 190995
Grupo: Nivelacion

Puno - Perú
2025

1 Ejercicio:

I) Ejercicio:

El precio de una vivienda (P) depende linealmente del área construida (A) y puede expresarse como P = mA+b, donde m es el costo por metro cuadrado y b representa costos fijos.

Codigo en Python:

```
import numpy as np
import matplotlib.pyplot as plt

# Definir los par metros
costo_m2 = 1500
costos_fijos = 50000

# Definir la funci n
def precio_vivienda(area):
    return costo_m2 * area + costos_fijos

# Generar datos ( reas  construidas)
areas = np.arange(50, 251, 10)  #  reas  de 50 a 250 m  , en
    incrementos de 10

# Calcular precios
precios = precio_vivienda(areas)

# Graficar
plt.plot(areas, precios)
plt.xlabel(" rea  construida (m  )")
plt.ylabel("Precio de la vivienda")
plt.title("Relaci n entre  rea  y precio de vivienda")
plt.grid(True)
plt.show()

# Mostrar los datos en una tabla (opcional)
data = {'Area (m  )': areas, 'Precio': precios}
df = pd.DataFrame(data)
print(df)
```

II) Ejercicio:

La ganancia mensual (G) de un modelo depende linealmente del número de predicciones realizadas (N) como G = cN +b, donde c es la ganancia por predicción y b son ingresos fijos.

Codigo en Python:

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

```

```

# Definir los par metros
ganancia_prediccion = 50
ingresos_fijos = 1000

# Definir la funci n
def ganancia_mensual(predicciones):
    return ganancia_prediccion * predicciones + ingresos_fijos

# Generar datos (n mero de predicciones)
predicciones = np.arange(0, 501, 25)
# Calcular ganancias
ganancias = ganancia_mensual(predicciones)

# Graficar
plt.plot(predicciones, ganancias)
plt.xlabel("N mero de predicciones")
plt.ylabel("Ganancia mensual")
plt.title("Relaci n entre predicciones y ganancia mensual")
plt.grid(True)
plt.show()

# Mostrar datos en tabla (opcional)
data = {'Predicciones': predicciones, 'Ganancia': ganancias}
df = pd.DataFrame(data)
print(df)
```

III) Ejercicio:

El tiempo total de procesamiento (T) en un algoritmo depende linealmente del tamaño de los datos (D), expresado como $T = kD + c$, donde k es el tiempo por unidad de datos y c es un tiempo constante de configuración.

Codigo en Python:

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Par metros del modelo
tiempo_por_unidad = 0.002
tiempo_configuracion = 0.1

# Definir la funci n
def tiempo_procesamiento(tama o_datos):
    return tiempo_por_unidad * tama o_datos + tiempo_configuracion

# Generar datos
tama os_datos = np.arange(100, 10001, 500)

# Calcular tiempos de procesamiento
tiempos = tiempo_procesamiento(tama os_datos)

# Graficar
plt.plot(tama os_datos, tiempos)
```

```
plt.xlabel("Tama o de los datos")
plt.ylabel("Tiempo de procesamiento (segundos)")
plt.title("Tiempo de procesamiento vs. Tama o de datos")
plt.grid(True)
plt.show()

# Mostrar datos en tabla
data = {'Tama o de datos': tama os_datos , 'Tiempo (seg)': tiempos}
df = pd.DataFrame(data)
print(df)
```

IV) Ejercicio:

El costo total (C) para almacenar datos depende linealmente de la cantidad de datos almacenados (D), según C = pD+f, donde p es el costo por gigabyte y f son tarifas fijas.

Codigo en Python:

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Par metros del modelo
costo_por_gb = 2
tarifas_fijas = 10

# Definir la funci n
def costo_almacenamiento(datos_gb):
    return costo_por_gb * datos_gb + tarifas_fijas

# Generar datos
datos_gb = np.arange(0, 101, 10)

# Calcular costos
costos = costo_almacenamiento(datos_gb)

# Graficar
plt.plot(datos_gb, costos)
plt.xlabel("Cantidad de datos (GB)")
plt.ylabel("Costo total")
plt.title("Costo de almacenamiento vs. Cantidad de datos")
plt.grid(True)
plt.show()

# Mostrar datos en tabla
data = {'Datos (GB)': datos_gb, 'Costo': costos}
df = pd.DataFrame(data)
print(df)
```

V) Ejercicio:

La medición calibrada (M) de un sensor depende linealmente de la medición en crudo (R) como M = aR+b, donde a es el factor de ajuste y b es un desplazamiento constante.

Codigo en Python:

```python
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Par metros del modelo
factor_ajuste = 1.2
desplazamiento = 5

# Definir la funci n
def medicion_calibrada(medicion_cruda):
    return factor_ajuste * medicion_cruda + desplazamiento

# Generar datos
mediciones_crudas = np.arange(0, 51, 5)

# Calcular mediciones calibradas
mediciones_calibradas = medicion_calibrada(mediciones_crudas)

# Graficar
plt.plot(mediciones_crudas, mediciones_calibradas)
plt.xlabel("Medici n en crudo")
plt.ylabel("Medici n calibrada")
plt.title("Medici n calibrada vs. Medici n en crudo")
plt.grid(True)
plt.show()

# Mostrar datos en tabla
data = {'Crudo': mediciones_crudas, 'Calibrado': mediciones_calibradas}
df = pd.DataFrame(data)
print(df)
```

VI) Ejercicio:

El tiempo de respuesta promedio (T) de un servidor depende linealmente del número de solicitudes simultáneas (S) como T = mS+b, donde m es el tiempo incremental por solicitud y b es el tiempo base.

Codigo en Python:

```python
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Par metros del modelo
tiempo_incremental = 0.01
tiempo_base = 0.05

# Definir la funci n
def tiempo_respuesta(solicitudes):
    return tiempo_incremental * solicitudes + tiempo_base

# Generar datos
solicitudes = np.arange(1, 101, 5)
```

```
15
16 # Calcular tiempos de respuesta
17 tiempos_respuesta = tiempo_respuesta(solicitudes)
18
19 # Graficar
20 plt.plot(solicitudes, tiempos_respuesta)
21 plt.xlabel("N mero de solicitudes simult neas")
22 plt.ylabel("Tiempo de respuesta promedio (segundos)")
23 plt.title("Tiempo de respuesta vs. N mero de solicitudes")
24 plt.grid(True)
25 plt.show()
26
27 # Mostrar datos en tabla
28 data = {'Solicitudes': solicitudes, 'Tiempo (seg)': tiempos_respuesta}
29 df = pd.DataFrame(data)
30 print(df)
```

VII) Ejercicio:

Los ingresos (I) de una plataforma dependen linealmente del número de suscriptores (S) como I = pS+b, donde p es el ingreso promedio por suscriptor y b son ingresos adicionales.

Codigo en Python:

```
1  import numpy as np
2  import matplotlib.pyplot as plt
3  import pandas as pd
4
5  # Par metros del modelo
6  ingreso_promedio = 10
7  ingresos_adicionales = 500
8
9  # Definir la funci n
10 def ingresos_plataforma(suscriptores):
11     return ingreso_promedio * suscriptores + ingresos_adicionales
12
13 # Generar datos
14 suscriptores = np.arange(0, 1001, 100)
15
16 # Calcular ingresos
17 ingresos = ingresos_plataforma(suscriptores)
18
19 # Graficar
20 plt.plot(suscriptores, ingresos)
21 plt.xlabel("N mero de suscriptores")
22 plt.ylabel("Ingresos")
23 plt.title("Ingresos de la plataforma vs. N mero de suscriptores")
24 plt.grid(True)
25 plt.show()
26
27 # Mostrar datos en tabla
28 data = {'Suscriptores': suscriptores, 'Ingresos': ingresos}
29 df = pd.DataFrame(data)
30 print(df)
```

VIII) Ejercicio:

La energía consumida (E) depende linealmente del número de operaciones realizadas (O) como E = kO+b, donde k es la energía consumida por operación y b es la energía base para encender el sistema..

Codigo en Python:

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Par metros del modelo
energia_por_operacion = 0.005
energia_base = 10

# Definir la funci n
def energia_consumida(operaciones):
    return energia_por_operacion * operaciones + energia_base

# Generar datos
operaciones = np.arange(0, 10001, 500)

# Calcular energ a consumida
energia = energia_consumida(operaciones)

# Graficar
plt.plot(operaciones, energia)
plt.xlabel("N mero de operaciones")
plt.ylabel("Energ a consumida (julios)")
plt.title("Energ a consumida vs. N mero de operaciones")
plt.grid(True)
plt.show()

# Mostrar datos en tabla
data = {'Operaciones': operaciones, 'Energ a (julios)': energia}
df = pd.DataFrame(data)
print(df)
```

IX) Ejercicio:

El número de likes (L) en una publicación depende linealmente del número de seguidores (F) como L = mF +b, donde m es la proporción promedio de interacción y b es un nivel base de likes.

Codigo en Python:

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Par metros del modelo
proporcion_interaccion = 0.1
likes_base = 5
```

```
8
9  # Definir la funci n
10 def numero_likes(seguidores):
11     return proporcion_interaccion * seguidores + likes_base
12
13 # Generar datos
14 seguidores = np.arange(0, 10001, 500)
15
16 # Calcular n mero de likes
17 likes = numero_likes(seguidores)
18
19 # Graficar
20 plt.plot(seguidores, likes)
21 plt.xlabel("N mero de seguidores")
22 plt.ylabel("N mero de likes")
23 plt.title("N mero de likes vs. N mero de seguidores")
24 plt.grid(True)
25 plt.show()
26
27 # Mostrar datos en tabla
28 data = {'Seguidores': seguidores, 'Likes': likes}
29 df = pd.DataFrame(data)
30 print(df)
```

X) Ejercicio:

El costo total (C) para entrenar un modelo de machine learning depende linealmente del número de iteraciones (I) como C = pI +c, donde p es el costo por iteración y c son costos iniciales.

Codigo en Python:

```
1  import numpy as np
2  import matplotlib.pyplot as plt
3  import pandas as pd
4
5  # Par metros del modelo
6  costo_por_iteracion = 0.02
7  costos_iniciales = 5
8
9  # Definir la funci n
10 def costo_entrenamiento(iteraciones):
11     return costo_por_iteracion * iteraciones + costos_iniciales
12
13 # Generar datos
14 iteraciones = np.arange(0, 10001, 500)
15
16 # Calcular costo total
17 costos = costo_entrenamiento(iteraciones)
18
19 # Graficar
20 plt.plot(iteraciones, costos)
21 plt.xlabel("N mero de iteraciones")
22 plt.ylabel("Costo total")
23 plt.title("Costo de entrenamiento vs. N mero de iteraciones")
```

```
plt.grid(True)
plt.show()

# Mostrar datos en tabla
data = {'Iteraciones': iteraciones, 'Costo': costos}
df = pd.DataFrame(data)
print(df)
```

2 Codigo QR GitHub: