Con un editor de código basado en la web, aprenda los conceptos básicos de Python y ponga en práctica sus conocimientos al codificar un proyecto de inversión corto de Bitcoin.

Si desea utilizar su entorno de desarrollo para codificar, descargue Python en el sitio web oficial; que le brinda un IDE (entorno de desarrollo integrado) para codificar.

https://www.python.org/downloads/

Os proporcionamos las maquetas de varios archivos que tendréis que completar para realizar todas las tareas indicadas.

Objetivos:

Demostrar conocimientos sobre el lenguaje de programación de secuencias de comandos Python en los siguientes campos:

- Variables
- Bucles
- Funciones
- Estructuras de datos
- If
- Archivos

Utilizará el editor de código (por ejemplo VSCode) para completar ejercicios.

En programación, la sintaxis es importante ya que describe la estructura de un lenguaje de programación válido. En términos simples, la sintaxis le dice a la computadora cómo leer código usando reglas que controlan la estructura de los símbolos, la puntuación y las palabras de un lenguaje de programación

Archivos:

BITCOIN.PY

En este proyecto, creará un programa que te diga cuando el valor de tu Bitcoin cae por debajo de $ 30,000.

Necesitaras:

- Crear una función para convertir Bitcoin a €

- Si su Bitcoin cae por debajo de 30,000€, imprima un mensaje.

Puede suponer que 1 Bitcoin vale 40,000€

```
investment_in_bitcoin = 1.2
bitcoin_to_usd = 40000

# 1) write a function to calculate bitcoin to €

# 2) use function to calculate if the investment is below $30,000
```

SHIPPING.PY

En este proyecto, creará un programa que calcula el coste total de la cesta de la compra de un cliente, incluido el envío.

- Si un cliente gasta más de 100€, obtiene envío gratis

- Si un cliente gasta <100€, el coste de envío es de 1.20€ por kg de peso de las canastas

Imprima el coste total de la cesta del cliente (incluido el envío) para completar este ejercicio.

```
customer_basket_cost = 34
customer_basket_weight = 44

# Write if statement here to calculate the total cost
```

SCRIPT.PY

Es el script inicial del proyecto, en principio imprimiremos un mensaje para comprobar el buen funcionamiento

```
# Write your python code here
print("Roses are #ff0000 Violets are #0000ff why my code´s working I haven´t a clue")
```

TAREA 1: INICIANDO NUESTRO ENTORNO

RESPONDA LAS SIGUIENTES PREGUNTAS

- Ejecute lo que se encuentra actualmente lo que se encuentra actualmente en el archivo script.py

TAREA 2 HOLA MUNDO

Para comenzar, creemos un programa simple que genere texto.

```
# This is an example comment
print("Hello World")
```

Como puede ver en el bloque de código de ejemplo anterior, es solo una línea (que se muestra en la línea 2), y cuando ejecutamos este código, generará el texto Hello World.

Analicemos esto. En el ejemplo, la línea 1 es un comentario, una línea que comienza con un símbolo de hashtag (#) y no la ejecuta la computadora. El programador (usted) escribe un comentario para ayudar a otras personas que leen el código a comprender lo que está sucediendo.

Podemos controlar lo que se envía a la pantalla usando la instrucción print (). Todo lo que esté dentro del paréntesis () se generará. Sin embargo, debido a que estamos imprimiendo una cadena (más sobre los tipos de datos más adelante en esta sala), tenemos que ponerlos entre comillas ""

Tenga en cuenta que los ejemplos proporcionados son para Python3.

RESPONDA LAS SIGUIENTES PREGUNTAS:

- En el editor de código, imprima "Hola mundo". Pero en un color determinado, en este caso verde ¿Cómo lo solucionaría?

TAREA 3 OPERADORES MATEMÁTICOS

Veamos ahora los operadores matemáticos y cómo se pueden aplicar a Python. Como una calculadora, existen operaciones como sumar, restar, multiplicar y dividir; usando Python, podemos codificar nuestra calculadora; después de todo, la programación es simplemente escribir reglas para que la computadora las siga dadas las entradas y condiciones específicas. La siguiente tabla muestra las diferentes operaciones.

Operator	Syntax	Example
Addition	+	1 + 1 = 2
Subtraction	-	5 - 1 = 4
Multiplication	*	10 * 10 = 100
Division	/	10 / 2 = 5
Modulus	%	10 % 2 = 0
Exponent	**	5**2 = 25 (5^2)

Ahora que conocemos los operadores matemáticos básicos, pasemos a los operadores de comparación; estos juegan un papel importante en Python y se desarrollarán cuando veamos los bucles y las declaraciones if. Estos operadores se utilizan para evaluar la condición de un programa en un estado particular.

Symbol	Syntax
Greater than	>
Less than	<
Equal to	==
Not Equal to	!=
Greater than or equal to	>=
Less than or equal	<=

RESPONDA LAS SIGUIENTES PREGUNTAS

- En el editor de código, imprima el resultado de 21 + 43., es decir, en el código, escriba lo siguiente: print (21 + 43) ¿Cómo es la sintaxis realizada?
- Imprime el resultado de 142 - 52. ¿Cómo es la sintaxis realizada?
- Imprime el resultado de 10 * 342. ¿Cómo es la sintaxis realizada?
- Imprime el resultado de 5 al cuadrado. ¿Cómo es la sintaxis realizada?

TAREA 4: VARIABLES Y TIPOS DE DATOS

Las variables le permiten almacenar y actualizar datos en un programa de computadora. Tiene un nombre de variable y almacena datos con ese nombre.

```
food = "ice cream"
money = 2000
```

En el ejemplo anterior, tenemos 2 variables. El nombre de la variable "comida" almacena la cadena (palabras) helado, mientras que otra variable llamada "dinero" almacena un número (2000).

Las variables son poderosas ya que puede cambiarlas a lo largo de su programa. El siguiente ejemplo establece la variable de edad en 30, luego aumentamos esta variable de edad en 1, lo que hace que los datos de la variable final sean 31. Siéntase libre de copiar y pegar esto en el editor, ejecutar el código y ver su resultado.

```
age = 30
age = age + 1
print(age)
```

Observe, en la línea 2, la forma en que actualizamos una variable, a la izquierda, y tenemos el nombre de variable ya creado "edad" seguido del operador =. A la derecha, tenemos a qué estamos configurando la variable; en nuestro caso, la variable edad (que actualmente está establecida en 30) se incrementa en 1.

Hablemos de tipos de datos, que es el tipo de datos que se almacenan en una variable. Puede almacenar texto o números y muchos otros tipos. Los tipos de datos que debe conocer son:

• **`String`**: se utiliza para combinaciones de caracteres, como letras o símbolos.

• **`Integer`**: números enteros

• **`Float`**: números que contienen puntos decimales o para fracciones

• **`Boolean`**: se utiliza para datos restringidos a las opciones Verdadero o Falso.

• **`List`**: serie de diferentes tipos de datos almacenados en una colección

String Float Integer Boolean List

Title	Rating	Times Viewed	Favorite	Seen By
Star Wars	9.8	13	True	Alice, Bob
Matrix	8.5	23	False	Charlie
Indiana Jones	6.1	3	False	Daniel, Evie

RESPONDA LAS SIGUIENTES PREGUNTAS

- En el archivo script.py, cree una variable llamada altura y establezca su valor inicial en 200.
- En una nueva línea, agregue 50 a la variable de altura.
- En otra línea nueva, imprima el valor de la altura. ¿Cómo es la sintaxis realizada?

TAREA 5 OPERADORES LÓGICOS Y BOOLEANOS

Los operadores lógicos permiten realizar asignaciones y comparaciones y se utilizan en pruebas condicionales (como declaraciones if).

Logical Operation	Operator	Example
Equivalence	==	if x == 5
Less than	<	if x < 5
Less than or equal to	<=	if x <= 5
Greater than	>	if x > 5
Greater than or equal to	>=	if x >= 5

Los operadores booleanos se utilizan para conectar y comparar relaciones entre declaraciones. Como una declaración if, las condiciones pueden ser verdaderas o falsas.

Boolean Operation	Operator	Example
Both conditions must be true for the statement to be true	**AND**	if x >= 5 **AND** x <= 100 Returns TRUE if x is a number between 5 and 100

Only one condition of the statement needs to be true	**OR**	if x == 1 **OR** x == 10 Returns TRUE if X is 1 or 10
If a condition is the opposite of an argument	**NOT**	if **NOT** y Returns TRUE if the y value is False

Veamos algunos ejemplos de código de Python:

```
a = 1
if a == 1 or a > 10:
     print("a is either 1 or above 10")
name = "bob" hungry = True
if name == "bob" and hungry == True:
     print("bob is hungry")
else if name == "bob" and not hungry:
     print("Bob is not hungry")
else: # If all other if conditions are not met
     print("Not sure who this is or if they are hungry")
```

RESPONDA LAS SIGUIENTES PREGUNTAS

Lea la sección anterior y cree su propio script.py con diferentes operadores lógicos y condiciones para que dependiendo de tu edad puedas comer una cantidad de helado sin quedarte con hambre y sin dinero. La especificaciones iniciales son las siguientes:

- Tu dinero inicial son 2000.
- Para satisfacer el hambre tiene que estar entre un rango del 85% y 100%.
- El precio inicial del Helado es de 100 y se incrementa un 20% cada vez que compres un helado.
- El porcentaje de hambre depende de la edad ya que será el porcentaje inicial de tu hambre (Es decir si tienes 26 años un 26%).

TAREA 6 EL PROYECTO SHIPPING

El uso de "declaraciones if" permite a los programas tomar decisiones. Permitieron que un programa eligiera una decisión en función de una condición. A continuación se muestra un ejemplo de cómo se puede usar una declaración if para determinar la sección de código (qué declaración de impresión) usar.

```
if age < 17:
    print('You are NOT old enough to drive')
else:
    print('You are old enough to drive')
```

En el ejemplo, si tiene menos de 17 años, el programa mostrará el texto "NO tiene la edad suficiente para conducir"; sin embargo, si tiene más de 17 años, el programa mostrará "Tiene edad suficiente para conducir". Dependiendo de una condición (en este ejemplo, es la variable de edad), el programa ejecutará diferentes secciones de código.

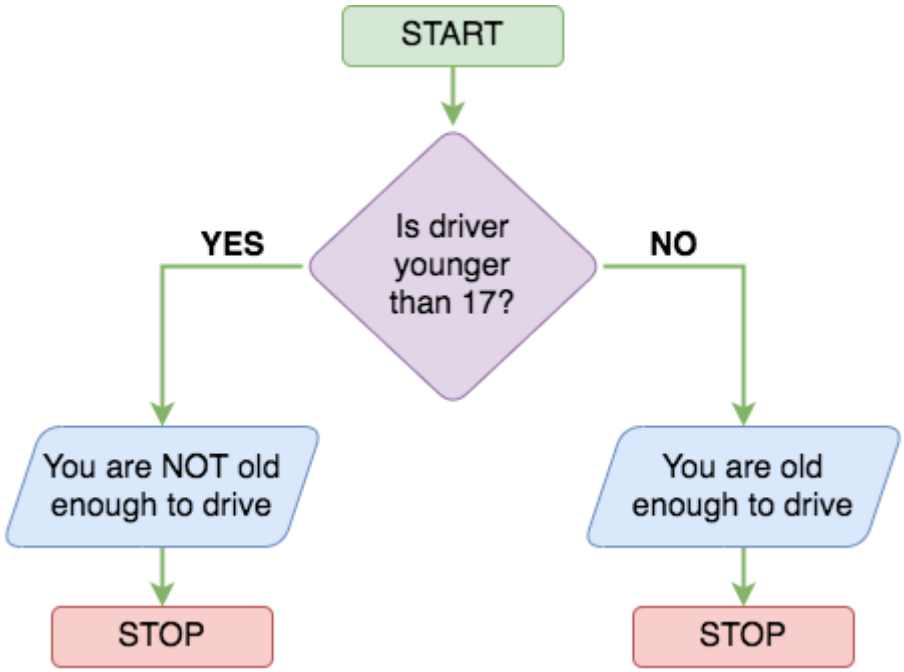
Hay algunos componentes clave que observamos en nuestro ejemplo de código anterior:

• La palabra clave if indica el comienzo de la instrucción if, seguida de un conjunto de condiciones.

• La instrucción if solo se ejecuta si la condición (o conjuntos de condiciones) es verdadera. En nuestro ejemplo, tiene menos de 17 años; si esa condición es verdadera (la edad es superior a 17), se ejecuta el código dentro de la instrucción if. Según el ejemplo, si no se cumplen determinadas condiciones, el programa puede utilizar de forma predeterminada el código de ejecución que se muestra en la parte else de la sentencia if.

• Dos puntos: marca el final de la instrucción if.

• Tenga en cuenta la sangría. Todo lo que esté sangrado después de los dos puntos se considera parte de la instrucción if, que ejecutará el programa.



Los diagramas de flujo son imprescindibles en la programación y será algo que uses mucho. Para ello te recomiendo que mires esta plataforma https://www.figma.com/ por ahora te servirá para realizar el

diagrama de flujo en la siguiente actividad, pero esta herramienta es muy potente y contiene muchas funcionalidades que te animo a que descubras

RESPONDA LAS SIGUIENTES PREGUNTAS

En este ejercicio, codificarás una pequeña aplicación que calcula y genera el costo de envío para un cliente en función de cuánto ha gastado.

- Introduce el código de shipping.py en el editor de código y realiza las instrucciones para completar esta tarea. Además, cree el diagrama de flujo en la herramienta Figma
- Una vez que haya escrito la aplicación en la pestaña shipping.py del editor de código, aparecerá un texto cuando compilas y ejecutas el código, que es la respuesta a esta pregunta. ¿Cuál es?
- En shipping.py, en la línea 12 (cuando use la sugerencia del editor de código), cambie la variable customer_basket_cost a 101 y vuelva a ejecutar su código. Obtendrá una nueva respuesta (si el costo total es correcto según su código); la bandera es la respuesta a esta pregunta. ¿Cuál es?

TAREA 7 BUCLES

En programación, los bucles permiten a los programas iterar y realizar acciones varias veces. Hay dos tipos de bucles, bucles for y while.

BUCLES WHILE

Comencemos observando cómo estructuramos un bucle while. Podemos hacer que el ciclo se ejecute indefinidamente o (similar a una instrucción if) determinar cuántas veces debe ejecutarse el ciclo en función de una condición.

```
i = 1
while i <= 10:
    print(i)
    i = i + 1
```

Este bucle while se ejecutará 10 veces, dando salida al valor de la variable i cada vez que itera (bucles). Analicemos esto:

- La variable i se establece en 1
- La instrucción while especifica dónde debe comenzar el inicio del ciclo.
- Cada vez que se repite, comenzará en la parte superior (emitiendo el valor de i)
- Luego pasa a la siguiente línea del ciclo, lo que aumenta el valor de i en 1
- Luego (como no hay más código para que el programa se ejecute), va a la parte superior del ciclo, comenzando el proceso de nuevo.
- El programa seguirá en bucle hasta que el valor de la variable i sea mayor que 10

BUCLES FOR

Un bucle for se utiliza para iterar sobre una secuencia, como una lista. Las listas se utilizan para almacenar varios elementos en una sola variable y se crean utilizando corchetes (ver más abajo). Aprendamos con el siguiente ejemplo:

```
websites = ["facebook.com", "google.com", "amazon.com"]
for site in websites:
    print(site)
```

Este bucle for que se muestra en el bloque de código anterior, se ejecutará 3 veces, dando salida a cada sitio web en la lista. Analicemos esto:

- La variable de lista llamada sitios web almacena 3 elementos
- El bucle itera a través de cada elemento, imprimiendo el elemento
- El programa deja de repetirse cuando ha pasado por cada elemento del circuito.

Para ofrecer un escenario del mundo real, puede crear un programa que verifique si un sitio web está en línea o si un artículo está en stock. Recorrería la lista de sitios web, agregaría funcionalidad dentro del bucle para verificar el sitio web y generaría los resultados. La sala "Python for Pentesters" le muestra cómo usar Python para enumerar un objetivo, construir un keylogger, escanear una red y más.

En Python, también podemos iterar a través de un rango de números usando la función de rango. A continuación se muestra un ejemplo de código Python que imprimirá los números del 0 al 4. En programación, 0 es a menudo el número inicial, por lo que contar hasta 5 es de 0 a 4 (pero tiene 5 números: 0, 1, 2, 3 y 4)

```
for i in range(5):
    print(i)
```

RESPONDA LAS SIGUIENTES PREGUNTAS

- En el editor de código, vuelva a utilizar el archivo "script.py" y codifique un bucle que genere todos los números del 0 al 50.
- Después elija un numero aleatorio para asignárselo a la edad para nuestro cálculo de hambre, además genere un diagrama de flujo con figma de nuestro script.py

TAREA 8 INTRODUCCIÓN A LAS FUNCIONES.PROYECTO BITCOIN

A medida que los programas comienzan a hacerse más grandes y complejos, parte de su código será repetitivo, escribiendo el mismo código para hacer los mismos cálculos, y aquí es donde entran las funciones. Una función es un bloque de código que se puede llamar en diferentes lugares en su programa.

Podría tener una función para realizar un cálculo, como la distancia entre dos puntos en un mapa o un texto con formato de salida basado en ciertas condiciones. Tener funciones elimina el código repetitivo, ya que el propósito de la función se puede usar varias veces a lo largo de un programa.

```
def sayHello(name):
```

```
    print("Hello " + name + "! Nice to meet you.")

sayHello("ben") # Output is: Hello Ben! Nice to meet you
```

Hay algunos componentes clave que podemos observar en esta función:

• La palabra clave def indica el comienzo de una función. La función va seguida de un nombre que define el programador (y es un parámetro de función). En nuestro ejemplo, es sayHello.

• Después del nombre de la función hay un par de paréntesis () que contiene valores de entrada, datos que podemos pasar a la función. En nuestro ejemplo, es un nombre.

• Dos puntos: marca el final del encabezado de la función.

En la función, observe la sangría. De manera similar a las declaraciones if, cualquier cosa después de los dos puntos que tenga sangría se considera parte de la función.

Una función también puede devolver un resultado, consulte el bloque de código a continuación:

```
def calcCost(item):
    if(item == "sweets"):
        return 3.99
    elif (item == "oranges"):
        return 1.99
    else:
        return 0.99

spent = 10
spent = spent + calcCost("sweets")
print("You have spent:" + str(spent))
```

Si llamamos a la función calcCost y pasamos "dulces" como parámetro del elemento, la función devolverá un número decimal (flotante). En el código anterior, tomamos una variable llamada gastada y agregamos el costo de "dulces" a través de la función calcCost; cuando llamamos a calcCost, devolverá el número 3.99.

RESPONDA LAS SIGUIENTES PREGUNTAS

Ha invertido en Bitcoin y desea escribir un programa que le indique cuándo el valor de Bitcoin cae por debajo de un valor particular en euros. Asegúrese de que su función de Python tenga el nombre bitcoinToEuros y utilice los nombres de parámetro bitcoin_amount y bitcoin_value_euros.

- En el editor de código, utilice el código bitcoin.py. Escriba una función llamada bitcoinToEuros con dos parámetros: bitcoin_amount, la cantidad de Bitcoin que posee y bitcoin_value_euros,

el valor de bitcoin en Euros. La función debe devolver euros_value, que es su valor de bitcoin en Euros (para calcular esto, en la función, multiplica la variable bitcoin_amount por la variable bitcoin_value_euros y devuelve el valor). El inicio de la función debería verse así:
    - def bitcoinToEuros (monto_bitcoin, valor_bitcoin_euros):
    - Una vez que haya escrito la función bitcoinToEuros, úsela para calcular el valor de su Bitcoin en Euros, y luego cree una declaración if para determinar si el valor cae por debajo de 30,000€; si lo hace, envíe un mensaje para alertarlo (a través de una declaración impresa).
- 1 Bitcoin ahora vale 25,000€. En el editor de código en la línea 14, actualice el valor de la variable bitcoin_to_euros a 25000€ y vea si su programa Python reconoce que su inversión está por debajo del umbral de 30,000€.

TAREA 9: ARCHIVOS

En Python, puede leer y escribir archivos. Es común escribir un script e importarlo o exportarlo desde un archivo; ya sea como una forma de almacenar la salida de su secuencia de comandos o para importar una lista de cientos de sitios web desde un archivo para enumerar. Vamos a sumergirnos directamente en un ejemplo:

```
f = open("file_name", "r")
print(f.read())
```

Para abrir el archivo, usamos la función incorporada open (), y el parámetro "r" significa "leer" y se usa mientras leemos el contenido del archivo. La variable tiene un método read () para leer el contenido del archivo. También puede utilizar el método readlines () y recorrer cada línea del archivo; útil si tiene una lista donde cada elemento está en una nueva línea. En el ejemplo anterior, el archivo está en la misma carpeta que la secuencia de comandos de Python; si estuviera en otro lugar, necesitaría especificar la ruta completa del archivo.

También puede crear y escribir archivos. Si está escribiendo en un archivo existente, primero abra el archivo y use la "a" en la función de abrir después de la llamada de nombre de archivo (que significa anexar). Si está escribiendo en un archivo nuevo, use "w" (escribir) en lugar de "a". Vea los ejemplos a continuación para mayor claridad:

```
f = open("demofile1.txt", "a") # Append to an existing file
f.write("The file will include more text..")
f.close()

f = open("demofile2.txt", "w") # Creating and writing to a new
file
f.write("demofile2 file created, with this content in!")
f.close()
```

Observe que usamos el método close () después de escribir en un archivo; esto cierra el archivo para que no se pueda escribir más en el archivo (dentro del programa).

RESPONDA LAS SIGUIENTES PREGUNTAS

- En el editor de código, cree un archivo que se llame flag.txt en el mismo conjunto de nuestro ejercicio y escriba el código Python en nuestro script.py para leer el archivo flag.txt. ¿Cuál es el resultado?

TAREA 10 IMPORTACIONES EN PYTHON

En Python, podemos importar bibliotecas, que son una colección de archivos que contienen funciones. Piense en importar una biblioteca como funciones de importación que puede usar y que ya se han escrito para usted. Por ejemplo, hay una biblioteca de "fechas" que le da acceso a cientos de funciones diferentes para cualquier cosa relacionada con la fecha y la hora.

```
import datetime
current_time = datetime.datetime.now()
print(current_time)
```

Importamos otras bibliotecas utilizando la palabra clave import. Luego, en Python, usamos el nombre de la biblioteca de esa importación para hacer referencia a sus funciones. En el ejemplo anterior, importamos fecha y hora, luego accedemos al método .now () llamando a library_name.method_name (). Copie y pegue el ejemplo anterior en el editor de código.

Aquí hay algunas bibliotecas populares que pueden resultarle útiles en la creación de scripts como pentester:

• Solicitud: biblioteca HTTP simple.

• Scapy: envía, huele, disecciona y falsifica paquetes de red
https://scapy.readthedocs.io/en/latest/introduction.html

• Pwntools: una biblioteca de desarrollo CTF y exploits.

https://docs.pwntools.com/en/stable/

Muchas de estas bibliotecas ya están integradas en el lenguaje de programación; sin embargo, las bibliotecas escritas por otros programadores que aún no están instaladas en su máquina se pueden instalar usando una aplicación llamada pip, que es el administrador de paquetes de Python. Digamos que desea instalar la biblioteca "scapy" (que le permite crear sus propios paquetes en código y enviarlos a otras máquinas); lo instala primero ejecutando el comando pip install scapy, después de lo cual en su programa ahora puede importar la biblioteca scapy.

RESPONDA LAS SIGUIENTES PREGUNTAS

Lea la tarea y ejecute el código de ejemplo de Python en nuestro script.py ¿Cuál es el resultado?

TAREA 11 APLICANDO LO APRENDIDO. MONTECARLO PARA LA TOMA DE DECISIONES

Uso de la simulación de Monte Carlo para tomar decisiones en la vida real

Recientemente, me enfrenté a una decisión muy difícil de tomar. Tuve que elegir entre varias ofertas de trabajo, todas interesantes, por diferentes motivos. Después de un par de noches sin dormir, me di cuenta de una cosa: ¿por qué no usar las herramientas a mi disposición para ayudarme a tomar la decisión?

Deberá construir una simulación de Monte Carlo para ayudarlo a tomar la mejor decisión. Estaré comparando tres ofertas de trabajo diferentes.

La simulación de Monte Carlo es una técnica matemática que se utiliza para estimar todos los resultados posibles cuando se trata de incertidumbre en dichos resultados. La idea es sencilla.

Digamos que necesitas decidir entre dos caminos para volver a casa. El primero tarda en promedio 20 minutos. Tiene muy pocos semáforos y muy poco tráfico. Es muy probable que siempre llegue exactamente cuándo lo espera. El segundo tarda una media de 15 minutos. Sin embargo, a menudo hay mucho más tráfico, semáforos, etc. Hay mucha incertidumbre y puede llevarlo entre 10 y 45 minutos.

La simulación Monte Carlo construye un modelo de posibles resultados aprovechando una distribución de probabilidad. Al simular el experimento, digamos 10.000 veces, se hace una buena idea de lo riesgosas que son las distintas opciones. A continuación, puede decidir qué carretera funciona mejor para ti.

Planteamiento de la Idea

Supongamos que se trata de varias ofertas de trabajo. El desafío al tomar una decisión es que siempre existe cierta incertidumbre. Por ejemplo, no siempre sabes exactamente lo que harás, quiénes serán tus colegas, cómo este trabajo te ayudará a avanzar en tu carrera, etc.

Ahí es donde entra en juego Monte Carlo. Estos son los tres pasos para seguir para configurar el problema.

1. Definir todas las variables que intervienen en la toma de una decisión (salario, condiciones, ubicación, conciliación, etc.)

2. Defina la importancia (ponderación) de cada variable. En otras palabras, ¿qué es lo más importante para ti?

3. Para cada opción, defina, para cada variable, un rango de posibilidad en la escala que considere apropiada (de 10 en este ejemplo). Por supuesto, cuanto más amplio sea el rango, más incierto significa que estás con la calificación que le has dado a esta variable. El rango representa la distribución de probabilidad de la variable.

Por supuesto, las partes 2 y 3 requieren un enfoque más cualitativo. Las calificaciones se otorgan en función de su percepción de las cosas, su opinión. Lo que alguien define como un entorno de trabajo 10/10 es diferente para todos y no existe una forma universal fácil de cuantificar esto.

Sin embargo, lo interesante de este enfoque es que puede ayudarlo a descubrir algunos prejuicios que podría tener inconscientemente sobre las diversas opciones. De hecho, al calificar todas las variables por sí mismas, es posible que se dé cuenta de que una opción se destaca en la mayoría de los aspectos.

SOLUCIONES DE LOS CÓDIGOS:

SHIPPING.PY

```
"""
    In this project, you'll create a program that calculates the total
    cost of a customers shopping basket, including shipping.

    - If a customer spends over $100, they get free shipping
    - If a customer spends < $100, the shipping cost is $1.20 per kg
of the baskets weight

    Print the customers total basket cost (including shipping) to
complete this exercise.

    ===> You've redeemed a hint. Replace the X's with code to complete
this exercise.

"""

shipping_cost_per_kg = 1.20
customer_basket_cost = 34
customer_basket_weight = 44

if(customer_basket_cost >= X):
  print('Free shipping!')
else:
  shipping_cost = customer_basket_weight * shipping_cost_per_kg
  customer_basket_cost = X

print("Total basket cost including shipping is
" + str(customer_basket_cost))
```

BITCOIN.PY

```
"""
    In this project, you'll create a program that that tells
    you when the value of your Bitcoin falls below $30,000.

    You will need to:
    - Create a function to convert Bitcoin to €
    - If your Bitcoin falls below 30,000€ print a message.

    You can assume that 1 Bitcoin is worth 40,000€

    ===> You've redeemed a hint. Replace the _'s with code to complete
this exercise.

"""
```

```
investment_in_bitcoin = 1.2
bitcoin_to_euros = 40000

# 1) write a function to calculate bitcoin to euros
def bitcoinToEuros(_, _):
  usd_value = _ * _
  return usd_value

investment_in_usd = bitcoinToEuros(investment_in_bitcoin, bitcoin_to_u
sd)
if investment_in_euros <= _:
  print("Investment below 30,000€! SELL!")
else:
  print("Investment above 30,000€")
```

MONTECARLO.PY

```
import random

import seaborn as sns

import matplotlib.pyplot as plt


iterations=10000

variables=['work','salary','conditions','career_advancement','social_impact','job_se
curity','life_balance','location','atmosphere','travel']


def Monte_Carlo(grade):

    final_results=[]

    weight=[0.15,0.09,0.12,0.11,0.12,0.02,0.11,0.07,0.1,0.11]

    for n in range(iterations):

        results=[]

        for i in range(len(variables)):

            value = weight[i] * (random.uniform(grade[i][0], grade[i][1]))

            results.append(value)


        final_results.append(sum(results))

    return final_results


a =
Monte_Carlo([[4,9],[8.5,10],[5,9],[8.5,9.5],[3,7],[4,9],[3,8],[7.5,8],[5,9],[0,6]])
```

```
b =
Monte_Carlo([[5,10],[4,4],[7,9],[2,8],[6,9.5],[8.5,10],[8,10],[0,7],[3,9],[0,3]])
c =
Monte_Carlo([[4,7],[6,8],[6,9],[6.5,9],[2,6],[6.5,9],[5.5,9],[9.5,9.5],[5,9],[4,9]])


#Plot the results
fig = plt.figure(figsize=(10,6))
sns.distplot(a)
sns.distplot(b)
sns.distplot(c)
fig.legend(labels=['Job A','Job B','Job C'])
plt.title('Monte-Carlo Distributions')
plt.show()
```

- En la línea 6 es donde defino todas las variables que se tienen en cuenta.
- En la línea 10 es donde se asignan todos los pesos a las diferentes variables.
- La línea 14 es muy importante. Es donde se genera un valor aleatorio basado en el rango que ha proporcionado para esta variable.
- Las líneas 20 a 22 contienen las listas de listas, con el rango de calificación para cada variable. Por ejemplo, para A, la primera lista [4,9] significa que la work variable puede encontrarse entre una calificación de 4 y 9 en la simulación.