|  |  |  |
| --- | --- | --- |
| | GUÍA 3.1.5:Guía Complementaria Ejercicios Resueltos Listas | | |
| Sigla | Asignatura | Experiencia de Aprendizaje |
| FPY1101 | Fundamentos de Programación | EA2: Optimizando el código en Python |
| Tiempo | Modalidad de Trabajo | Indicadores de logro |
| 3 h | Individual | IL 3.1 al IL 3.3 |

|  |
| --- |
| **Código QR con relleno sólido**  **Antecedentes generales** |

## Esta guía tiene como objetivo conocer los aspectos generales en Python

### Ser una guía paso a paso para la construcción de programas en Python

### Todos los ejercicios tienen la solución incluida, pero antes de ver la solución, debes resolver por tu cuenta el ejercicio, de esa manera podrás reforzar y aprender. Las soluciones utilízalas para comparar con tus resultados, tomar nota o simplemente como revisión. debes ser consiente y responsable en tu autoaprendizaje.

### Debate con tu docente las respuestas obtenidas, si tienes dudas, recuerda dar aviso y argumentar, los programas no tienen soluciones únicas, teniendo derivaciones o mecanismos distintos de funcionamiento.

### Esta guía puede ser desarrollada en casa, o guiada por el docente, con el fin de seguir un paso a paso y comprender las explicaciones de la o el docente.

|  |
| --- |
| **Lista con relleno sólido Requerimientos para esta actividad** |

## Para el desarrollo de esta actividad deberás disponer de:

## Computador

## Visual Studio Code

|  |  |
| --- | --- |
| **Inteligencia artificial con relleno sólidoActividad** |  |

**EJERCICIO 1**

|  |  |
| --- | --- |
| **Enunciado** | El complejo educacional “Chile 2030”, desea realizar una aplicación computacional que le permita registrar en sus establecimientos los alumnos a sus cursos.  Por tal razón le ha solicitado que cree un programa que le permita a coordinación estudiantil registrar los alumnos que pertenezcan a un curso en particular.  Como prototipo, usted desarrolla un algoritmo que permite almacenar un número variable de alumnos a un curso, pero con un máximo de 30 por curso.  Construya el código que responda a los siguientes resultados: |
| **Algoritmo o resultado** |  |
| **Solución** | cursos = []  print("")  contador = 0  opcion = ""  while opcion != "n":    nombre = input("Ingrese el nombre del alumno: ")  direccion = input("Ingrese la dirección del alumno: ")  telefono = input("Ingrese el teléfono del alumno: ")  alumno = []  alumno = [nombre,direccion,telefono]  cursos.append(alumno)  contador += 1  if contador == 30:  opcion = "n"  print("")  else:  opcion = input("Desea agregar otro alumno (s/n): ")  print("")      for alumno in cursos:  print("Nombre: " ,alumno[0])  print("Dirección: ", alumno[1])  print("Teléfono: ", alumno[2])  print("")    print("") |
| **Feedback** | -Para poder utilizar una lista primero hay que crearla, es decir,  “cursos = []” y “alumno = []” no pueden faltar.  -Para agregar más alumnos a la lista “cursos”, se debe utilizar el método append.  Algoritmo Correcto:  cursos = []  print("")  contador = 0  opcion = ""  while opcion != "n":    nombre = input("Ingrese el nombre del alumno: ")  direccion = input("Ingrese la dirección del alumno: ")  telefono = input("Ingrese el teléfono del alumno: ")  alumno = []  alumno = [nombre,direccion,telefono]  cursos.append(alumno)  contador += 1  if contador == 30:  opcion = "n"  print("")  else:  opcion = input("Desea agregar otro alumno (s/n): ")  print("")      for alumno in cursos:  print("Nombre: " ,alumno[0])  print("Dirección: ", alumno[1])  print("Teléfono: ", alumno[2])  print("")    print("") |

**EJERCICIO 2**

|  |  |
| --- | --- |
| **Enunciado** | La empresa de juegos de azar “Lotín”, desea crea una aplicación móvil que genere sorteos rápidos dentro del mismo celular.  Para eso, le solicita crear un algoritmo que genere una lista de números de manera aleatoria donde si usted acierta, gana. Las reglas son las siguientes:   * Los números participantes son del 1 al 49 * El jugador debe elegir 7 * Se generarán 3 rondas de conjunto de números ganadores * Si el jugador acierta a una de las 3 ronda, gana   Programe las líneas de código faltantes |
| **Algoritmo o resultado** | import random    suerte = []  print(“”)  print(“Ingrese sus 7 números del a suerte”)  for numero in range(7):  Línea incógnita 1    print(“”)  print(“Usted ingresó los siguientes números: “, suerte)  Línea incógnita 2  lista = []  for turno in range(1,7+1) :  flag = True  while flag :  aleatorio = random.randint(1,49)  if lista.count(aleatorio) == 0 :  Línea incógnita 3  flag = False  print(“”)  print(“”)  print(f”Los números sorteados en la ronda {ronda+1} fueron: “)  Línea incógnita 4  print(numero)  print(“”)    contador = 0  for numero in suerte:  if lista.count(numero) == 1:  contador += 1    if contador == 7 :  print(“Hoy es su día de suerte. Ha ganado ¡!!!”)  for i in range(10):  print(“Eres una ganador!!!!!!!!!!”)  break  else:  print(“Lo siento, pero no has ganado en esta ronda”) |
| **Solución** | Línea incógnita 1: suerte.append(int(input(“Ingrese número: “)))  Línea incógnita 2: for ronda in range(3):  Línea incógnita 3: lista.append(aleatorio)  Línea incógnita 4: for numero in lista: |
| **Feedback** | -Para agregar un valor a una lista es requerido el método append.  -Para recorrer una lista con el ciclo for, no se debe colocar el número de elementos luego del nombre de la lista. Es decir, lista(7) es inválido.  -Dentro del ciclo for si se utiliza range con un parámetro, range se mueve desde 0 a n-1. Por lo tanto, en el ejercicio se generarían solo 2 ciclos.  Algortimo correcto:    import random    suerte = []  print(“”)  print(“Ingrese sus 7 números del a suerte”)  for numero in range(7):  suerte.append(int(input(“Ingrese número: “)))    print(“”)  print(“Usted ingresó los siguientes números: “, suerte)    for ronda in range(3):  lista = []  for turno in range(1,7+1) :  flag = True  while flag :  aleatorio = random.randint(1,49)  if lista.count(aleatorio) == 0 :  lista.append(aleatorio)  flag = False  print(“”)  print(“”)  print(f”Los números sorteados en la ronda {ronda+1} fueron: “)  for numero in lista:  print(numero)  print(“”)    contador = 0  for numero in suerte:  if lista.count(numero) == 1:  contador += 1    if contador == 7 :  print(“Hoy es su día de suerte. Ha ganado ¡!!!”)  for i in range(10):  print(“Eres una ganador!!!!!!!!!!”)  break  else:  print(“Lo siento, pero no has ganado en esta ronda”) |

**EJERCICIO 3**

|  |  |
| --- | --- |
| **Enunciado** | Un amigo que trabaja en el instituto de estadísticas, le solicita un programa que le permita ordenar los datos de un arreglo para asi calcular fácilmente la mediana.  Usted, con agrado le dice que si, puesto que le gustan mucho los desafíos. Entonces recuerda que en sus clases de programación, en una prueba salió el algoritmo de ordenamiento por burbuja y decide hacerlo con el.  Este algoritmo ordena los números presentes en un arreglo de menor a mayor. Funciona comparando cada elemento de la lista con su siguiente (siempre la comparación es de solo 2 elementos a la vez).  Si los elementos están ordenados avanza una posición. Si los elementos están al revés intercambia los valores de posición y luego avanza en una posición.  Repite el ciclo muchas veces hasta que termina.  A continuación, se presenta un ejemplo de un arreglo si orden y luego el resultado ordenado.  (Seleccione la sección faltante correcta) |
| **Algoritmo o** **resultado** | import random  arreglo =[]    for ciclo in range(50):  aleatorio = random.randint(1,1000)  arreglo.append(aleatorio)    print("")  print("Arreglo Original")  print(arreglo)  print("")    largo\_arreglo = len(arreglo)    Sección faltante      print("")  print("Arreglo Ordenado")  print(arreglo)    if largo\_arreglo%2 == 0:  mediana = (arreglo[int(largo\_arreglo/2)-1] + arreglo[int(largo\_arreglo/2)])/2  else:  mediana = (arreglo[round(int(largo\_arreglo/2))])  print("Mediana: ", mediana) |
| **Opción 1** | for i in range(largo\_arreglo - 1):  for j in range(largo\_arreglo - 1 - i):  if arreglo[j] < arreglo[j+1]:  temporal = arreglo[j]  arreglo[j] = arreglo[j+1]  arreglo[j+1] = temporal |
| **Opción 2** | for i in range(largo\_arreglo - 1):  for j in range(largo\_arreglo - 1 - i):  if arreglo[j] > arreglo[j+1]:  arreglo[j] = arreglo[j+1]  arreglo[j+1] = arreglo[j] |
| **Opción 3** | for i in range(largo\_arreglo - 1):  for j in range(largo\_arreglo - 1 - i):  if arreglo[j] < arreglo[j+1]:  arreglo[j] = arreglo[j+1]  arreglo[j+1] = arreglo[j] |
| **Opción 4 - Solución** | for i in range(largo\_arreglo - 1):  for j in range(largo\_arreglo - 1 - i):  if arreglo[j] > arreglo[j+1]:  temporal = arreglo[j]  arreglo[j] = arreglo[j+1]  arreglo[j+1] = temporal |
| **Feedback** | -El valor de arreglo[j] debe transferirse a una variable temporal, puesto que, de no ser así, ese valor será sobre escrito.  -Si la primera posición es mayor se hace el cambio. De lo contrario no se realiza ningún cambio del valor.  Algoritmo completo:  import random  arreglo =[]    for ciclo in range(50):  aleatorio = random.randint(1,1000)  arreglo.append(aleatorio)    print("")  print("Arreglo Original")  print(arreglo)  print("")    largo\_arreglo = len(arreglo)    for i in range(largo\_arreglo - 1):  for j in range(largo\_arreglo - 1 - i):  if arreglo[j] > arreglo[j+1]:  temporal = arreglo[j]  arreglo[j] = arreglo[j+1]  arreglo[j+1] = temporal      print("")  print("Arreglo Ordenado")  print(arreglo)    if largo\_arreglo%2 == 0:  mediana = (arreglo[int(largo\_arreglo/2)-1] + arreglo[int(largo\_arreglo/2)])/2  else:  mediana = (arreglo[round(int(largo\_arreglo/2))])  print("Mediana: ", mediana) |

**EJERCICIO 4**

|  |  |
| --- | --- |
| **Enunciado** | Interesado en el mundo de las acciones bursátiles, su jefe, desea implementar un algoritmo que le permita graficar la media móvil simple, para poder determinar las tendencias de las distintas acciones que sigue dentro del mercado.  La media móvil simple consiste en ir obteniendo los promedios parciales de cierto conjunto de valores. De esta manera se logra que los cambios de las curvas sean más suaves.  Por ejemplo, si se tienen 50 datos de precios, y se utiliza un periodo de 3 (se sacarán los promedios de a grupos de 3), el primer valor de la media móvil sería el promedio de los primeros 3 datos. El segundo valor de la media móvil será el promedio del 2do, 3ro y 4to valores de los datos y así sucesivamente.  Un ejemplo resultante se presenta a continuación. En azul los datos de los valores de las acciones y en naranjo los valores de la media móvil.  Construye el código faltante para calcular la media móvil simple en una serie de datos representada por la lista precios. Utilizando dos bucles anidados, el bucle externo se desplaza a través de la lista, definiendo ventanas de datos de longitud especificada por la variable periodos. En cada iteración, se suma el valor de cada elemento dentro de la ventana y se calcula el promedio. El resultado se redondea y se agrega a la lista media\_movil. Este proceso se repite para diferentes segmentos de la serie, generando una lista de medias móviles que suavizan las fluctuaciones a corto plazo en la serie temporal. |
| **Algoritmo o resultado** | Algortimo  import random  import matplotlib.pyplot as plt    fig, ax = plt.subplots()    precios = []  tiempo = []  media\_movil = []    for t in range(51):  precios.append(random.randint(1,20))  tiempo.append(t)    periodos = 3    Algoritmo faltante  for i in range(periodos-1):  primero = media\_movil[0]  media\_movil.insert(0,primero)      ax.plot(tiempo, precios)  ax.plot(tiempo, media\_movil)  ax.set\_title(“Media óvil: Precio vs Tiempo”)  ax.set\_xlabel(“Tiempo”)  ax.set\_ylabel(“Precio”)    plt.show() |
| **Solución** | for conteo in range(len(precios)-periodos+1):    promedio = 0  for item in range(periodos):  promedio += precios[item+conteo]  promedio = round(promedio / periodos)    media\_movil.append(promedio) |
| **Feedback** | -Un promedio se obtiene sumando los valores y luego dividiendo por la cantidad de estos. La variable promedio debe escribirse así:  promedio += precios[item+conteo]  Y no así:  promedio = precios[item+conteo]  La segunda alternativa está siempre reemplazando y no sumando los precios.  -La variable promedio debe hacerse cero cada vez “promedio = 0”, de lo contrario estará considerando el promedio anterior.  -Los promedios resultantes deben agregarse a las lista media\_movil. Para eso se requiere de la función append. Si esa función un nuevo valor no será incluido.  Algortimo correcto:  import random  import matplotlib.pyplot as plt    fig, ax = plt.subplots()    precios = []  tiempo = []  media\_movil = []    for t in range(51):  precios.append(random.randint(1,20))  tiempo.append(t)    periodos = 3  for conteo in range(len(precios)-periodos+1):    promedio = 0  for item in range(periodos):  promedio += precios[item+conteo]  promedio = round(promedio / periodos)    media\_movil.append(promedio)    for i in range(periodos-1):  primero = media\_movil[0]  media\_movil.insert(0,primero)      ax.plot(tiempo, precios)  ax.plot(tiempo, media\_movil)  ax.set\_title(“Media óvil: Precio vs Tiempo”)  ax.set\_xlabel(“Tiempo”)  ax.set\_ylabel(“Precio”)    plt.show() |

**EJERCICIO 5**

|  |  |
| --- | --- |
| **Enunciado** | El almacén ubicado cerca de su hogar requiere de un pequeño sistema informático que permita calcular las compras de los clientes.  Para esto, se deben crear la lista de los productos en conjunto con sus precios según el siguiente ejemplo:  productos = [“Tomate”, “Leche”,”Huevos”,”Harina”,”Levadura”, “Mantequilla”,”Queso”,”Vienesas”]  precios = [500,900,2500,1000,500,2300,4000,3000]  En este caso hay correspondencia directa de las ubicaciones dentro de la lista del producto y dentro de la lista de los precios. Es decir, el tomate vale 500, la leche vale 900, etc.  Para calcular el costo, se debe solicitar la cantidad de cada producto según se muestra en la imagen. El valor también se coloca dentro de una lista, en el mismo orden de la lista productos y precios.  Construya las línea de código faltantes |
| **Algoritmo o resultado** |  |
| **Solución** | productos = ["Tomate", "Leche","Huevos","Harina","Levadura", "Mantequilla","Queso","Vienesas"]  cantidad = []  precios = [500,900,2500,1000,500,2300,4000,3000]  total = 0    print("")  print("Lista de compras")  contador = 0  while contador < len(productos) :  print(productos[contador])  print(precios[contador])  cantidad.append(int(input("Ingrese cantidad: ")))  print("")  total += precios[contador] \* cantidad[contador]  contador += 1    print("")  print("Detalle pago:")  print("Total afecto: {:,}".format(total))  iva = total\*0.19  print("Iva: {:,}".format(iva))  total\_final = iva + total  print("Total a pagar: {:,}".format(total\_final))  print("") |
| **Feedback** | -Para poder utilizar el valor presente dentro de una lista se debe indicar el índice de la posición del valor dentro de la lista. Ejemplo: lista[índice]. Si una lista está sin el índice, no se puede obtener el valor particular.  -El acumulador total requiere de la estructura +=. Esta estructura simboliza a total = total + valor. Es decir, cada nueva suma se suma al resultado anterior.  -Lo usual es que, si estoy utilizando un contador, ese se incremente de iteración en iteración. Por lo tanto en la condición while, esta se ejecutará mientras contador no llegue al límite.  Algoritmo completo:      productos = ["Tomate", "Leche","Huevos","Harina","Levadura", "Mantequilla","Queso","Vienesas"]  cantidad = []  precios = [500,900,2500,1000,500,2300,4000,3000]  total = 0    print("")  print("Lista de compras")  contador = 0  while contador < len(productos) :  print(productos[contador])  print(precios[contador])  cantidad.append(int(input("Ingrese cantidad: ")))  print("")  total += precios[contador] \* cantidad[contador]  contador += 1    print("")  print("Detalle pago:")  print("Total afecto: {:,}".format(total))  iva = total\*0.19  print("Iva: {:,}".format(iva))  total\_final = iva + total  print("Total a pagar: {:,}".format(total\_final))  print("") |