

APELLIDO:	CALIFICACIÓN:
NOMBRE:	
DNI (registrado en SIU Guaraní):	
E-MAIL:	DOCENTE (nombre y apellido):
TEL:	
AULA:	

Duración del examen: 1:20h. Completar con **letra clara, mayúscula e imprenta**. El examen consta de 11 preguntas de opción múltiple. Cada pregunta tiene una y sólo una respuesta correcta.

Las respuestas deben completarse con una X en la siguiente matriz:

Opción	EJ. 1	EJ. 2	EJ. 3	EJ. 4	EJ. 5	EJ. 6	EJ. 7	EJ. 8	EJ. 9	EJ. 10	EJ. 11
1											
2											
3											
4											

¡ATENCIÓN! Las respuestas sólo se considerarán válidas si se encuentran en la matriz. De haber diferencias entre la opción seleccionada en el ejercicio y en la matriz, se considerará como válida la de la matriz.

Ejercicio 0108 - 1 punto			
¿Qué contiene <i>b</i> ? <pre>def organiza(n): if n%3==0 and n%2==0 or n<100: return True else: return False a=[15,-150,150,66] b=list(filter(organiza,a))</pre>			
1.	[15, -150, 150, 66]	X	1
2.	[-150, 66]		2
3.	[15, 66]		3
4.	[]		4

Ejercicio 0208 - 1 punto			
¿Cuál versión de la función ingreso() valida correctamente los datos de acceso a una cuenta de mail? Deben coincidir usuario y password <pre>def ingreso(usr): - - # users contiene dirección de mail:[password,usuario] users={'info@usina.com.ar':['X25fc230','joanaBur'], 'iro@hotmail.com':['Irma1211','rojas_27'], 'ergo@gmail.com':['joacoS01','infoGENERAL']} while not ingreso(users): print('Datos de Acceso Erróneos')</pre>			
1.	<pre>def ingreso(usr): usuario=input('Usuario: ') passw=input('Password: ').lower() resp=False for datos in usr: if usr[datos][0]==usuario: resp=True return resp</pre>		1

2.	<pre>def ingreso(usr): usuario=input('Usuario: ') passw=input('Password: ') resp=False for datos in usr: if usr[datos]==[passw,usuario]: resp=True return resp</pre>	X	2
3.	<pre>def ingreso(usr): usuario=input('Usuario: ') passw=input('Password: ') return usuario in usr or passw in usr</pre>		3
4.	<pre>def ingreso(usr): usuario=input('Usuario: ').lower() passw=input('Password: ').lower() return usuario in usr and passw in usr</pre>		4

Ejercicio 0308 - 1 punto			
¿Cuál versión de la función abreParaLeer() evita la interrupción del programa por un fallo de archivo no encontrado en el caso de que el archivo no se encuentre? Nota: Se espera que la función evite la parada del programa por ese fallo y eventualmente cree un archivo nuevo, sólo si no existe			
<pre>def abreParaLeer(arch): - - datos=abreParaLeer('uno.txt') - - datos.close()</pre>			
1.	<pre>def abreParaLeer(arch): a=open(arch,'r') try: return a except FileNotFoundError: a=open(arch,'w') a.close() a=open(arch,'r') return a</pre>		1
2.	<pre>def abreParaLeer(arch): try: a=open(arch,'r') return a except FileNotFoundError: print('No existe',arch)</pre>		2
3.	<pre>def abreParaLeer(arch): a=open(arch,'w') try: a=open(arch) return a except: a=open(arch,'w') a.close() a=open(arch,'r') return a</pre>		3
4.	<pre>def abreParaLeer(arch): try: a=open(arch,'r') return a except FileNotFoundError: a=open(arch,'w') a.close() a=open(arch,'r') return a</pre>	X	4

Ejercicio 0408 - 1 punto			
¿Cuál versión de la función guarda() genera un archivo personal.txt que contendrá únicamente los datos que le pasa el programa? El archivo debe quedar con el siguiente formato y contenido: <pre>pa,ANA PAZ ai,INÉS ALZA os,SERGIO ORTIZ</pre> <pre>def guarda(dicci,arch): - - nombres={'ana':'paz','inés':'alza','sergio':'ortiz'} guarda(nombres,'personal.txt')</pre>			
1.	<pre>def guarda(dicci,arch): dat=open(arch,'a',encoding='utf-8') for nom in dicci: inicio=dicci[nom][0].upper()+nom[0]+' ' dat.write(inicio+'\n'+nom+' '+dicci[nom]) dat.close()</pre>		2
2.	<pre>def guarda(dicci,arch): dat=open(arch,'W',encoding='utf-8') for nom in dicci: inicio=nom[0]+dicci[nom][1].upper()+', ' fin=nom+' '+dicci[nom].upper()+'\n' dat.write(inicio+'\n'+fin) arch.close()</pre>		3
3.	<pre>def guarda(dicci,arch): dat=open(arch,'w',encoding='utf-8') lineas=[] for nom in dicci: inicio=dicci[nom][0]+nom[0]+' ' fin=nom.upper()+ ' '+dicci[nom].upper() lineas.append(inicio+fin+'\n') dat.writelines(lineas) dat.close()</pre>	X	4
4.	<pre>def guarda(dicci,arch): dat=open(arch,'r',encoding='utf-8') lineas=[] for nom in dicci: inicio=nom[1]+nom[0]+' ' fin=nom.upper()+ ' '+nom[1].upper()+'\n' lineas.append(inicio+fin) dat.writelines(lineas) arch.close()</pre>		

Ejercicio 0508 - 1 punto

Dado el siguiente DataFrame *lluvias*:

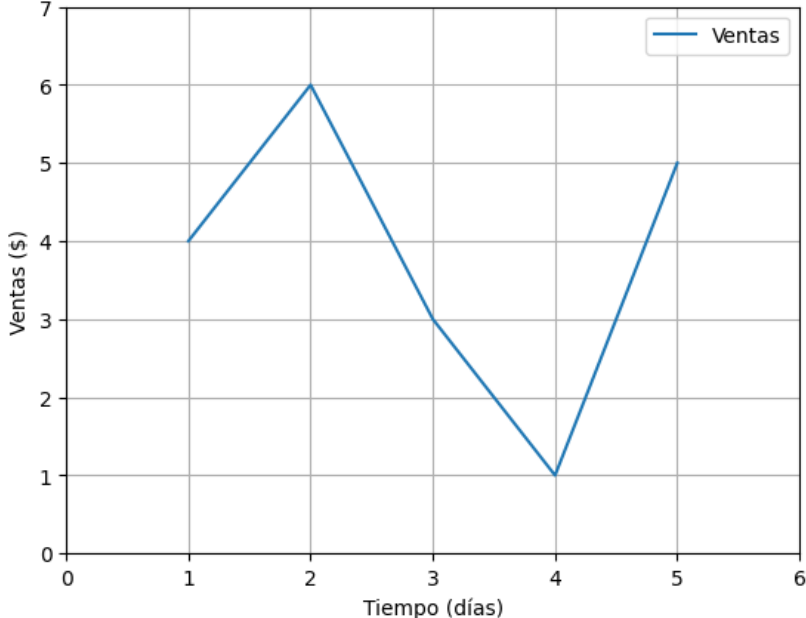
	localidad	mes	mm
0	azul	abril	65
1	charata	abril	96
2	azul	julio	30
3	azul	octubre	72
4	federación	enero	110
5	quitolipi	marzo	120
6	federación	marzo	115

Que contiene 7 filas y 3 columnas: localidad, mes y precipitación total registrada en mm (mm).

¿Qué instrucción produce la siguiente salida?			
<pre>mes abril 80.5 enero 110.0 julio 30.0 marzo 117.5 octubre 72.0</pre>			
1.	<code>lluvias.describe()</code>		1
2.	<code>lluvias.groupby('mes')['mm'].mean()</code>	X	2
3.	<code>lluvias.loc[1:3,['mes','mm']]</code>		3
4.	<code>lluvias.head(4)</code>		4

Ejercicio 0608 - 1 punto			
Dado el siguiente programa: <pre>def obtieneNom(t): return 'ana' in t.lower().split()[0] nombres=['ana López','emiliano SAL','LORENA ana báunes', 'analía Soto','ángelea FALCÓN', 'Luciana analía perez','Ana María Giménez'] resultado= . . . for nom in resultado: print(nom)</pre> Que produce la siguiente salida: ana López analía Soto Luciana analía perez Ana María Giménez >>> Qué instrucción debería ir en los puntos suspensivos? Nota: El argumento <i>key</i> permite pasarle a la función un criterio alternativo de comparación entre los elementos de la estructura. En este caso se comparan las versiones de los nombres en mayúsculas.			
1.	<code>list(reversed(nombres, key=obtieneNom))</code>		1
2.	<code>list(filter(obtieneNom, nombres))</code>	X	2
3.	<code>list(map(key=obtieneNom, nombres))</code>		3
4.	<code>nombres.sort(key=obtieneNom)</code>		4

Ejercicio 0708 - 1 punto			
Dado el siguiente programa: <pre>print('Elimina Elementos de') nombres=['julio','ana','inés','juan','lía'] print(nombres) sigue=True respSi=('s','si','sipi','sisi','oki','dale') while sigue: opc=input('Saca? (si/no) ') if opc.lower() in respSi: try: elimina=nombres.pop() except IndexError: sigue=False else: sigue=False</pre> Y los siguientes ingresos: <pre>Elimina Elementos de ['julio', 'ana', 'inés', 'juan', 'lía'] Saca? (si/no) s Saca? (si/no) S Saca? (si/no) SIPI Saca? (si/no) DALE Saca? (si/no) si Saca? (si/no) sisi</pre> ¿Qué contenido tendrá <i>nombres</i> al finalizar?			
1.	[]	X	1
2.	['julio', 'ana', 'inés', 'juan', 'lía']		2
3.	['juan', 'lía']		3
4.	['julio']		4

Ejercicio 0808 - 1 punto															
Gráfico de ventas <table><caption>Data points for 'Gráfico de ventas'</caption><tr><th>Tiempo (días)</th><th>Ventas (\$)</th></tr><tr><td>1</td><td>4</td></tr><tr><td>2</td><td>6</td></tr><tr><td>3</td><td>3</td></tr><tr><td>4</td><td>1</td></tr><tr><td>5</td><td>5</td></tr></table> ¿Cuál de las siguientes líneas de código NO SE CORRESPONDE con la figura?				Tiempo (días)	Ventas (\$)	1	4	2	6	3	3	4	1	5	5
Tiempo (días)	Ventas (\$)														
1	4														
2	6														
3	3														
4	1														
5	5														
1.	<code>ax.set_ylabel('Ventas (\$)')</code>		1												
2.	<code>ax.set_title("Gráfico de ventas")</code>		2												
3.	<code>ax.plot(x, y, label='Ventas (\$)')</code>	X	3												
4.	<code>ax.plot(x, y, label='Ventas')</code>		4												

Ejercicio 0908 - 1 punto			
Dado el siguiente programa <pre>archivo = open("archivo.txt", "r") lineas = archivo.readlines() archivo.close() for linea in lineas: campos = linea.split(";") nombre = tuple(campos[0].split(",")) direcccion = tuple(campos[1].split(",")) print(f"Nombre: {nombre[0]} {nombre[1]}") print(f"Dirección: {direcccion[0]} {direcccion[1]}")</pre> Cuando se ejecuta imprime <pre>Nombre: Luis Huergo Dirección: Paseo Colón 850 Nombre: Elisa Bachofen Dirección: Las Heras 2214</pre> ¿ Cúal es la estructura de archivo.txt ?			
1.	nombre apellido;calle altura;código postal localidad		1
2.	nombre,apellido;calle,altura;código postal,localidad	X	2
3.	nombre;apellido,calle;altura,código postal;localidad		3
4.	nombre,apellido,calle,altura,código postal,localidad		4

Ejercicio 1008 - 2 puntos			
¿Cómo queda el archivo instrucciones.txt luego de ejecutar el siguiente programa? <pre>def leer(arch): receta=open(arch,'r',encoding='utf-8') lineas=receta.readlines() receta.close() pasos=[] i=0 bandera=True while i<len(lineas) and bandera: if 'Preparación' in lineas[i]: bandera=False i+=1 if i<len(lineas): while i<len(lineas): pasos.append(lineas[i]) i+=1 return pasos def guarda(lista,arch): datos=open(arch,'w',encoding='utf-8') for i in range(len(lista)): desc=lista[i].strip('\n').capitalize() datos.write(str(i+1)+' - '+desc+'\n') datos.close() pasos=leer('receta.txt') guarda(pasos,'instrucciones.txt')</pre>			
Nota: El contenido de receta.txt es el siguiente: Ingredientes harina 200 gr fécula 300 gr yemas 3 unid manteca 200 gr azúcar 150 gr dulce de leche c/n esencia de vainilla 1 chdita ralladura de limón 1 chdita coco rallado c/n polvo de hornear 1 chdita Preparación cremar manteca con azúcar integrar las yemas saborizar incorporar secos estirar masa cortar discos hornear			
1.	Ingredientes harina 200 gr fécula 300 gr yemas 3 unid manteca 200 gr azúcar 150 gr dulce de leche c/n esencia de vainilla 1 chdita ralladura de limón 1 chdita coco rallado c/n polvo de hornear 1 chdita		1
2.	CREMAR MANTECA CON AZÚCAR INTEGRAR LAS YEMAS SABORIZAR INCORPORAR SECOS ESTIRAR MASA CORTAR DISCOS HORNEAR		2
3.	1 - Cremar manteca con azúcar 2 - Integrar las yemas 3 - Saborizar 4 - Incorporar secos 5 - Estirar masa 6 - Cortar discos 7 - Hornear	X	3
4.	Cremar manteca con azúcar Integrar las yemas Saborizar Incorporar secos Estirar masa Cortar discos Hornear		4

Ejercicio 1108 - 2 puntos

Dado el siguiente DataFrame *cobertura*:

	región	provincia	sucursales
0	cuyo	san juan	3
1	litoral	santa fé	0
2	cuyo	mendoza	5
3	patagonia	chubut	2
4	cuyo	san luis	1
5	pampa	buenos aires	8

Que contiene 6 filas y 3 columnas: región geográfica (región), provincia y cantidad de sucursales abiertas (sucursales).

¿Qué se muestra como resultado al ejecutar las siguientes instrucciones?

```
cobertura['región']=cobertura['región'].map({'cuyo':'centro','pampa':'centro'})
cobertura[(cobertura['región'].isnull()==False)]
```

1.	0 2 4 5	región cuyo cuyo cuyo pampa	sucursales 3 5 1 8		1	
2.	5	región centro	provincia buenos aires	sucursales 8		2
3.	0 2 4 5	región centro centro centro centro	provincia san juan mendoza san luis buenos aires	sucursales 3 5 1 8	X	3
4.	0 1 2 3 4 5	región centro litoral centro patagonia centro centro	provincia san juan santa fé mendoza chubut san luis buenos aires			4