

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 0107 - 1 punto			
¿Qué contiene <i>b</i> ? <pre>def organiza(n): if n%3==0 and n<100 and n%2==0: return True else: return False a=[15,-150,150,66] b=list(filter(organiza,a))</pre>			
1.	[66, -150]		1
2.	[-150, 66]	X	2
3.	[150, 66]		3
4.	[]		4

Ejercicio 0207 - 1 punto			
¿Cuál versión de la función ingreso() valida correctamente los datos de acceso a una cuenta de mail? Deben coincidir dirección de mail y password <pre>def ingreso(usr): def ingreso(usr): - - # users contiene dirección de mail:password users={'info@usina.com.ar': 'X25fc230', 'iro@hotmail.com': 'Irma1211', 'ergo@gmail.com': 'joacoS01'} while not ingreso(users): print('Datos de Acceso Erróneos')</pre>			
1.	<pre>def ingreso(usr): dccMail=input('Mail: ') clave=input('Password: ') if dccMail in usr and clave==usr[dccMail]: return True</pre>	X	1
2.	<pre>def ingreso(usr): dccMail=input('Mail: ') clave=input('Password: ') if clave==usr[dccMail].lower(): return True</pre>		2

3.	<pre>def ingreso(usr): dccMail=input('Mail: ') clave=input('Password: ') i=0 resp=False for mail in usr: if dccMail==mail: if usr[i]==clave: resp=True i+=1 return resp</pre>		3
4.	<pre>def ingreso(usr): dccMail=input('Mail: ').lower() clave=input('Password: ') resp=True if dccMail in usr: if usr[dccMail]==clave: resp=False return resp</pre>		4

Ejercicio 0307 - 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): try: a=open(arch,'r') return a except FileNotFoundError: print('No existe el archivo')</pre>		1
2.	<pre>def abreParaLeer(arch): a=open(arch,'w') a.close() try: a=open(arch,'r') except FileNotFoundError: a=open(arch,'w') return a</pre>		2
3.	<pre>def abreParaLeer(arch): try: a=open(arch) except FileNotFoundError: a=open(arch,'w') a.close() a=open(arch) return a</pre>	X	3
4.	<pre>def abreParaLeer(arch): a=open(arch,'r+') try: a=open(arch,'a') except FileNotFoundError: a=open(arch,'r') return a</pre>		4

Ejercicio 0407 - 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: ana paz,AP inés alza,IA sergio ortiz,SO <pre>def guarda(dicci,arch): - - nombres={'ana':'paz','inés':'alza','sergio':'ortiz'} guarda(nombres,'personal.txt')</pre>	

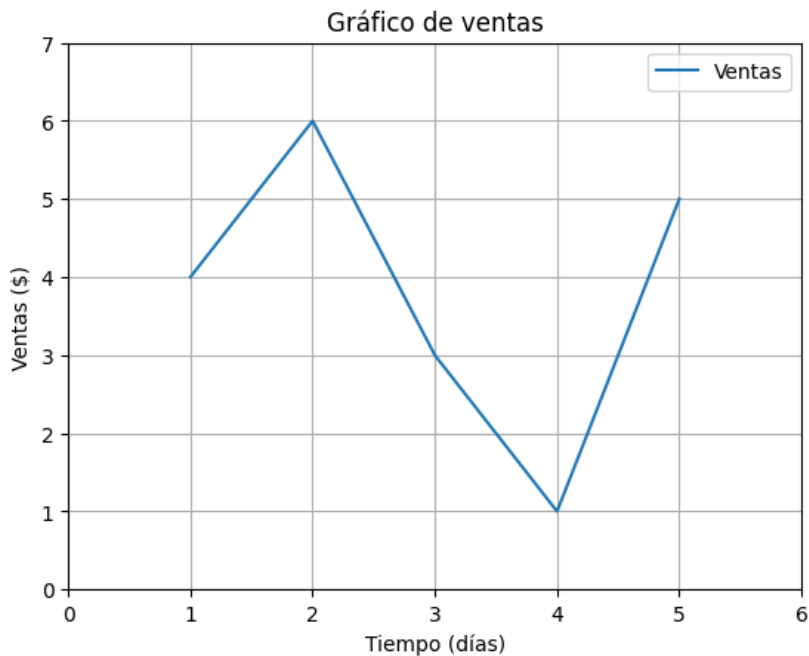
1.	<pre>def guarda(dicci,arch): dat=open(arch,'r',encoding='utf-8') lineas=[] for nom in dicci: ape=dicci[nom] lin=ape[0]+nom[0]+' '+ape.upper()+' '+nom.upper() lineas.append(lin) dat.writelines(lineas) dat.close()</pre>		2
2.	<pre>def guarda(dicci,arch): dat=open(arch,'w',encoding='utf-8') for i in range(len(dicci)): ape=dicci[i] dat.write(ape[0]+ape[1]+' '+ape[0].upper()+ape[1].upper()) arch.close()</pre>		3
3.	<pre>def guarda(d,arch): datos=open(arch,'a',encoding='utf-8') for nom in d: ape=d[nom] datos.write(nom[0]+ape[1]+' '+nom+ape) datos.close()</pre>		4
4.	<pre>def guarda(dicci,arch): dat=open(arch,'w',encoding='utf-8') for nom in dicci: ape=dicci[nom] dat.write(nom+' '+ape+' '+nom[0].upper()+ape[0].upper()+'\n') dat.close()</pre>	X	

Ejercicio 0507 - 1 punto																																			
Dado el siguiente DataFrame <i>lluvias</i>:																																			
<table><tr><th></th><th>localidad</th><th>mes</th><th>mm</th></tr><tr><td>0</td><td>azul</td><td>abril</td><td>65</td></tr><tr><td>1</td><td>charata</td><td>abril</td><td>96</td></tr><tr><td>2</td><td>azul</td><td>julio</td><td>30</td></tr><tr><td>3</td><td>azul</td><td>octubre</td><td>72</td></tr><tr><td>4</td><td>federación</td><td>enero</td><td>110</td></tr><tr><td>5</td><td>quitolipi</td><td>marzo</td><td>120</td></tr><tr><td>6</td><td>federación</td><td>marzo</td><td>115</td></tr></table>					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
	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?																																			
<table><tr><th></th><th>localidad</th><th>mes</th><th>mm</th></tr><tr><td>0</td><td>azul</td><td>abril</td><td>65</td></tr><tr><td>2</td><td>azul</td><td>julio</td><td>30</td></tr><tr><td>3</td><td>azul</td><td>octubre</td><td>72</td></tr></table>					localidad	mes	mm	0	azul	abril	65	2	azul	julio	30	3	azul	octubre	72																
	localidad	mes	mm																																
0	azul	abril	65																																
2	azul	julio	30																																
3	azul	octubre	72																																
1.	<code>lluvias[(lluvias['localidad']=='azul')]</code>	X	1																																
2.	<code>lluvias.head()</code>		2																																
3.	<code>lluvias.sum()</code>		3																																
4.	<code>lluvias.loc[lluvias.index[[0,2,3]],['mm']].sum()</code>		4																																

Ejercicio 0607 - 1 punto			
Dado el siguiente programa:<pre>def obtieneNom(t): return 'ez' in t.lower().split() [-1] nombres=['ana López','emiliano SAL','LORENA ana báunes', 'analía Soto','ángelea FALCÓN', 'Luciana analía p��rez','Ana Mar��a Gim��nez'] nombres.sort() resultado= . . . for nom in resultado: print(nom) Que produce la siguiente salida: Ana Mar��a Gim��nez Luciana anal��a p��rez ana L��pez >>> ��Qu�� instrucci��n deber��a ir en los puntos suspensivos? <u>Nota:</u> El argumento <i>key</i> permite pasarle a la funci��n un criterio alternativo de comparaci��n entre los elementos de la estructura. En este caso se comparan las versiones de los nombres en may��sculas.</pre>			
1.	<code>list(map(obtieneNom,nombres))</code>		1
2.	<code>reversed(nombres)</code>		2
3.	<code>obtieneNom(nombres)</code>		3
4.	<code>list(filter(obtieneNom,nombres))</code>	X	4

Ejercicio 0707 - 1 punto			
Dado el siguiente programa:<pre>print('Elimina Elementos de') nombres=['julio','ana','in��s','juan','l��a'] 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 Y los siguientes ingresos: Elimina Elementos de ['julio', 'ana', 'in��s', 'juan', 'l��a'] Saca? (si/no) dale Saca? (si/no) SIpi Saca? (si/no) nono ��Qu�� contenido tendr�� <i>nombres</i> al finalizar?</pre>			
1.	<code>[]</code>		1
2.	<code>['julio', 'ana', 'in��s', 'juan', 'l��a']</code>		2
3.	<code>['ana', 'in��s', 'juan', 'l��a']</code>		3
4.	<code>['julio', 'ana', 'in��s']</code>	X	4

Ejercicio 0807 - 1 punto



¿Cuál de las siguientes líneas de código NO SE CORRESPONDE con la figura?

1.	<code>ax.set_xlim(0, 8)</code>	X	1
2.	<code>ax.grid()</code>		2
3.	<code>ax.set_ylabel('Ventas (\$)')</code>		3
4.	<code>ax.set_title("Gráfico de ventas")</code>		4

Ejercicio 0907 - 1 punto

Dado el siguiente programa, ¿ cómo queda el archivo luego de cada ejecución ?

```
archivo=open('agenda.txt')
contactos=archivo.readlines()
archivo.close()

apellido=input('Ingresá Apellido del nuevo Contacto: ')
nombre=input(f'Nombre de {apellido}: ')
celular=input(f'Celular de {nombre} {apellido}: ')
nuevo=apellido +' '+ nombre +', ' + celular +'\n'
contactos.extend(nuevo)
archivo=open('agenda.txt', 'w')
archivo.writelines(contactos)
archivo.close()
```

1.	Con una línea		1
2.	Se agrega una nueva línea	X	2
3.	Se reemplaza la primer línea		3
4.	Ninguna de las anteriores		4

Ejercicio 1007 - 2 puntos ¿Cómo queda el archivo sePesa.txt luego de ejecutar el siguiente programa? <pre>def leer(arch): receta=open(arch,encoding='utf-8') instrucciones=receta.readlines() receta.close() ingrePesar=list(filter(soloPeso,instrucciones)) return ingrePesar def soloPeso(txt): return 'gr\n' in txt def guarda(lista,arch): datos=open(arch,'w',encoding='utf-8') for ingre in lista: desc=ingre.strip('gr\n').upper() datos.write(desc+'gr\n') datos.close() aPesar=leer('receta.txt') guarda(aPesar,'sePesa.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.	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.	HARINA 200 gr FÉCULA 300 gr MANTECA 200 gr AZÚCAR 150 gr	X	2
3.	harina 200 gr fécula 300 gr manteca 200 gr azúcar 150 gr		3
4.	YEMAS 3 DULCE DE LECHE c/n ESENCIA DE VAINILLA 1 RALLADURA DE LIMÓN 1 COCO RALLADO c/n POLVO DE HORNEAR 1		4

Ejercicio 1107 - 2 puntos																															
Dado el siguiente DataFrame <i>cobertura</i>:																															
<table><tr><td></td><td>región</td><td>provincia</td><td>sucursales</td></tr><tr><td>0</td><td>cuyo</td><td>san juan</td><td>3</td></tr><tr><td>1</td><td>litoral</td><td>santa fé</td><td>0</td></tr><tr><td>2</td><td>cuyo</td><td>mendoza</td><td>5</td></tr><tr><td>3</td><td>patagonia</td><td>chubut</td><td>2</td></tr><tr><td>4</td><td>cuyo</td><td>san luis</td><td>1</td></tr><tr><td>5</td><td>pampa</td><td>buenos aires</td><td>8</td></tr></table>				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	
	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?																															
<pre>cobertura=cobertura.drop(1) cobertura['región'].value_counts()</pre>																															
1.	cuyo 3 patagonia 1 pampa 1 santa fé 1		1																												
2.	cuyo 3 patagonia 1 pampa 1	X	2																												
3.	cuyo 9 patagonia 2 pampa 8		3																												
4.	cuyo 3		4																												