

PENSAMIENTO COMPUTACIONAL (90)

UBAXXI

TEMA 8

EXAMEN: SEGUNDO PARCIAL

APELLIDO:	CALIFICACIÓN:
NOMBRE:	
DNI (registrado en SIU Guaraní):	
E-MAIL:	NOTA Y FIRMA DOCENTE (no rellenar)
TEL:	
AULA:	

Duración del examen: 1:30h.

- ✓ Escribir claramente el nombre en todas las páginas.
- ✓ El examen consta de 9 preguntas de opción múltiple.
- ✓ Cada pregunta tiene una y sólo una respuesta correcta.
- ✓ Las respuestas seleccionadas deben consignarse en la siguiente matriz de opciones.
- ✓ **Sólo se considerarán las respuestas anotadas en la matriz.**
- ✓ Las preguntas de la 1 a la 5 inclusive permiten acumular 1 punto (si son correctas), de la 6 a la 9 cada una acumula 2 puntos o 0.
- ✓ La nota final se calcula de acuerdo a la siguiente función:

Puntos	1 o 2	3 o 4	5 o 6	7	8	9	10	11	12	13
Nota	1	2	3	4	5	6	7	8	9	10

Matriz de Respuestas

	Ej 1 1 Pto	Ej 2 1 Pto	Ej 3 1 Pto	Ej 4 1 Pto	Ej 5 1 Pto	Ej 6 2 Ptos	Ej 7 2 Ptos	Ej 8 2 Ptos	Ej 9 2 Ptos	
1										1
2										2
3										3
4										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 esta última.



Talón de Control para el Alumno

	Ej 1 1 Pto	Ej 2 1 Pto	Ej 3 1 Pto	Ej 4 1 Pto	Ej 5 1 Pto	Ej 6 2 Ptos	Ej 7 2 Ptos	Ej 8 2 Ptos	Ej 9 2 Ptos	
1										1
2										2
3										3
4										4

Ej 01 T8 – 1 Pto				
Para el siguiente DataFrame mascotas:				
	Nombre	Clase	Edad	Dueño
0	Pipo	loro	7	Juan
1	Tato	gato	5	Jimena
2	Kempes	gato	10	Lara
3	Tasso	perro	2	Hernán
¿Qué instrucción produce la siguiente salida?				
	Nombre	Clase	Edad	Dueño
0	Pipo		loro	7 Juan
1	mascotas.drop(columns=['Clase'])			1
2	mascotas[(mascotas['Clase']!='gato') & (mascotas['Edad']>3)]			X 2
3	mascotas['Clase'].value_counts()			3
4	mascotas['Clase'].map({'loro': 'LORAZO', 'gato': 'MIAU', 'perro': 'ROPE'})			4

Ej 02 T8 – 1 Pto				
El archivo Cientes.txt contiene lo siguiente:				
Salazar srl.;20-33456876-2				
Artelac;30-36925778-4				
Novallen;30-124445879-3				
starluz sa.;45896985				
¿Qué contenido tendrá el archivo nuevo.txt al finalizar la ejecución del siguiente programa?				
<pre> arch=open('Cientes.txt','r') lineas=arch.readlines() arch.close() arch=open('nuevo.txt','w') lineas.reverse() for lin in lineas: arch.write(lin) arch.close() </pre>				
1	30-36925778-4 30-124445879-3 20-33456876-2 45896985			1
2	Artelac			2
3	Salazar srl.;20-33456876-2 Novallen;30-124445879-3			3
4	starluz sa.;45896985 Novallen;30-124445879-3 Artelac;30-36925778-4 Salazar srl.;20-33456876-2			X 4

Ej 03 T8 – 1 Pto			
¿Qué muestra el siguiente programa?			
<pre>def edita(p): p=p.upper() devuelve=p.replace('AN','') return devuelve nom1=['Luciana','Lucía','lucas','JUAN'] nom2=list(map(edita,nom1)) print(*nom2)</pre>			
Notas:			
Usar un * antes de una lista en un print provoca que se muestran los elementos de la lista separados por blanco, sin los corchetes ni las comas			
Ej:			
print(*[1,2,0]) -> 1 2 0			
1	luciANa lucía lucas juAN		1
2	LUCIANA JUAN		2
3	True False False True		3
4	LUCIA LUCÍA LUCAS JU	X	4

Ej 04 T8 – 1 Pto			
¿Cuál es la función adecuada para el siguiente programa?			
<pre>def telegrama(...): - - - cambios={'anuncio':'', 'de ':' ', 'lluvias fuertes':'TORMENTAS', 'últimas horas':'final', 'primeras':'inicio', '':'', 'para':'', 'probable':'', 'las':'', 'y':'-', 'la':'', 'con':'',} txt='Anuncio de lluvias fuertes con caída probable de granizo para las últimas horas de la tarde y primeras de mañana' print(telegrama(cambios,txt))</pre>			
La salida del programa debe ser:			
TORMENTAS , caída granizo final tarde - inicio mañana			
1	<pre>def telegrama(): t.lower() for cl in d: t=t.replace(cl,d[cl]) return ''</pre>		1
2	<pre>def telegrama(d,t): txt=t.lower() for cl in d: txt=txt.replace(cl,d[cl]) return txt</pre>	X	2
3	<pre>def telegrama(d,t): t=t.lower() for cl in d: t.replace(d[cl],cl)</pre>		3
4	<pre>def telegrama(d,t,txt): txt=t.lower() for cl in range(len(d)): txt=t.replace(cl,d[cl]) return txt</pre>		4

Ej 05 T8 – 1 Pto			
<p>¿Cuál es el programa correcto que asegura el ingreso de un número par menor que 80? El programa debe insistir en el ingreso hasta asegurarse que la variable num contenga un número par (entero mayor que cero y divisible por 2) y menor a 80.</p>			
1	<pre>while True: try: num=int(input('Ingresa un numero par: ')) if num in range(2,80,2): sigue=True else: print('No cumple ser par o menor a 80') sigue=False except ValueError: print('Debe ser un entero') sigue=False print(num)</pre>		1
2	<pre>intenta=True while intenta: try: num=int(input('Ingresa un numero par: ')) if num in range(2,80) and num%2==0: intenta=False else: print('No cumple ser par o menor a 80') except ValueError: print('Debe ser un entero') print(num)</pre>	X	2
3	<pre>try: num=int(input('Ingresa un numero par: ')) if num in range(2,80,2): num=2 else except: print('No cumple ser par o menor a 80') num=1 except FileNotFoundError: print('Debe ser un entero') num=1 print(num)</pre>		3
4	<pre>intenta=True while intenta: try: num=int(input('Ingresa un numero par: ')) if num not in range(2,80,2): print('No cumple ser par o menor a 80') except ValueError: print('Debe ser un entero') intenta=True print(num)</pre>		4

Ej 06 T8 – 2 Ptos			
<p>¿Qué muestra por pantalla el siguiente programa?</p> <pre>def funcion(n): resp=False decimales=n-int(n) primeros3=int(decimales*1000) if primeros3<200: resp=True return resp numeros=[3.22,7.002,8.01,15.66,1.375] nuevaLista=list(filter(funcion,numeros)) print(nuevaLista)</pre>			
1	[3.22, 7.002, 8.01, 15.66, 1.375]		1
2	[]		2
3	[7.002, 8.01]	X	3
4	[15.66]		4

Ej 07 T8 – 2 Ptos			
<p>¿Cuál diccionario valores funcionará correctamente para el siguiente programa?</p> <pre>valores={...} #indicar cuál funcionará para la salida de abajo txt='Se abonaron 22800 por 451 de varilla de hierro de 1/2. Costo envío: 1110' for valor in valores: txt=txt.replace(valor,valores[valor]) print(txt)</pre> <p>La salida del programa debe ser:</p> <p>Se abonaron veintidos mil ochocientos pesos por cuatrocientos cincuenta y un kg de varilla de hierro de media pulgada. Costo envío: mil ciento diez pesos</p>			
1	{ 1:['media pulgada'], 2:['cuatrocientos cincuenta y un kg'], 3:['mil ciento diez pesos'], 4:['veintidos mil ochocientos pesos'] }		1
2	{'1/2':'media pulgada', '451': 'cuatrocientos cincuenta y un kg', '1110':'mil ciento diez pesos', '22800':'veintidos mil ochocientos pesos'}	X	2
3	{'media pulgada': '1/2', 'cuatrocientos cincuenta y un kg': '451', 'mil ciento diez pesos': '1110', 'veintidos mil ochocientos pesos': '22800'}		3
4	{':1}		4

Ej 08 T8 – 2 Ptos			
<p>¿Qué muestra el siguiente programa?</p> <pre>def sabores(m): categorias={1:'agua',2:'agua',3:'crema', 4:'crema',5:'agua',6:'chocolate', 7:'chocolate',8:'crema', 9:'crema',10:'crema',11:'sin tacc', 12:'sin tacc'} return categorias[m] in ('crema','sin tacc') gustosHela={1:'Limón',2:'Frutilla',3:'Vainilla', 4:'Pistacho',5:'Frambuesa',6:'Suizo', 7:'Italiano',8:'Americana',9:'Amarena', 10:'Del Cielo',11:'Dce de Leche',12:'Melón'} elegidos=[5,6,9,1,11] comprar=list(filter(sabores,elegidos)) for m in comprar: print(gustosHela[m])</pre>			
1	PISTACHO VAINILLA		1
2	melón-pistacho-suizo-dce de leche		2
3	Amarena Dce de Leche	X	3
4	Frambuesa Suizo Limón		4

Ej 09 T8 – 2 Ptos			
<p>Para el siguiente DataFrame mascotas:</p> <pre> Nombre Clase Edad Dueño 0 Pipo loro 7 Juan 1 Tato gato 5 Jimena 2 Kempes gato 10 Lara 3 Tasso perro 2 Hernán</pre> <p>¿Qué salida produce la siguiente instrucción?</p> <pre>mascotas.groupby('Clase')['Edad'].max()</pre>			
1	Dueño 1 Jimena 2 Lara		1
2	Dueño Nombre 0 Juan Pipo 1 Jimena Tato 2 Lara Kempes 3 Hernán Tasso		2
3	Nombre Clase Edad Dueño 2 Kempes gato 10 Lara 3 Tasso perro 2 Hernán		3
4	Edad Clase gato 10 loro 7 perro 2 dtype: int64	X	4