PENSAMIENTO COMPUTACIONAL (90) .UBAXXI

EXAMEN: RECUPERATORIO SEGUNDO PARCIAL – PRIMER CUATRIMESTRE 2025 **TEMA 2**

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**

Consignar con una X (cruz) la respuesta elegida en cada ejercicio en la siguiente matriz:

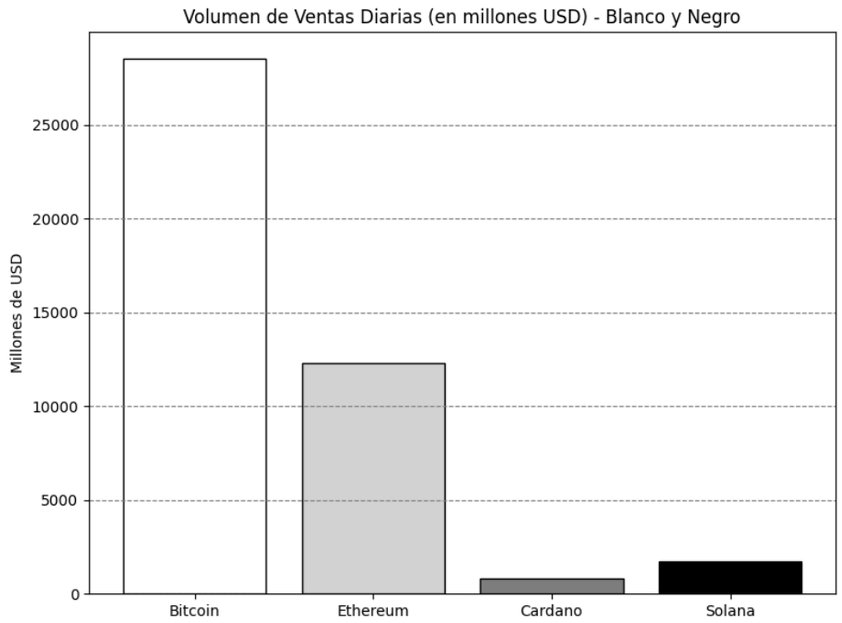
	Ej 1 1 Pto	Ej 2 1 Pto	Ej 3 1 Pto	Ej 4 1 Pto	Ej 5 1 Pto	Ej 6 2 Ptos	Ej 7 2 Ptos	Ej 8 2 Ptos	Ej 9 2 Ptos
A									
B									
C									
D									

Talón de Control para el Alumno (Tema ~~1~~ 2)

	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
A									
B									
C									
D									

Ejercicio 1 – Tema 2			**1 Pto**
Cuál de los siguientes diccionarios tiene **tipos** diferentes de claves (tipo int, float, str, tuple, bool …)?			
A	`{1:2, 23:'5.1', 5:13}`		**A**
B	`{True:'V', 1:'uno', 4:'cuatro', False:'F'}`	**X**	**B**
C	`{'3':0, '2':66, '5':3.45, '1':[-2,5]}`		**C**
D	`{('3','5'):44, ('1', '0'):2}`		**D**

Ejercicio 2 – Tema 2			**1 Pto**
¿Cuál de las siguientes instrucciones debería usar (junto a otras) para **generar** un gráfico como el siguiente con matplotlib?			



A	`plt.bar(criptos, ventas, color=colores, edgecolor='black')`	**X**	**A**
B	`plt.title('SOL')`		**B**
C	`plt.ylabel('BTC-ETH-ADA-SOL')`		**C**
D	`plt.pie(sizes, labels=labels, colors=colors, autopct='%1.1f%%', startangle=90, wedgeprops={'edgecolor': 'black'})`		**D**

Ejercicio 3 – Tema 2	**1 Pto**

Dado el Dataframe ***clientes***:

```
  apellido  edad       ciudad   provincia
0    Antón  50.0      Chajarí  Entre Ríos
1  Khunter   NaN    Concordia  Entre Ríos
2  Ortúzar  44.0   La Carlota     Córdoba
3    Gómez  23.0      Chajarí  Entre Ríos
4     Arce  35.0       La Paz  Entre Ríos
5    Alves  31.0  Salsipuedes     Córdoba
6    Alves  41.0         None        None
7     Díaz  72.0   Tafí Viejo     Tucumán
```

¿Cuál de las siguientes instrucciones debería usar para **producir** la siguiente salida?

```
provincia
Córdoba       31.0
Entre Ríos    23.0
Tucumán       72.0
```

A	`clientes.head(2)`		**A**
B	`clientes.sort_values(by=['provincia','ciudad'], ascending=[True,False])`		**B**
C	`clientes.groupby('provincia')['edad'].min()`	**X**	**C**
D	`clientes['ciudad'].value_counts()`		**D**

Ejercicio 4 – Tema 2	**1 Pto**

Dado el archivo ***vtas-suc1.txt*** que contiene las ventas mensuales del primer semestre de la sucursal 1:

vtas-suc1.txt

```
ene,69.5
feb,33.0
mar,20.0
abr,10.0
may,0
jun,0
```

¿Cuál es la sentencia que falta en el programa? Al finalizar quedará 1 archivo adicional: **vtas-semestre.txt**, que contendrá una línea con el total de las ventas del semestre de la sucursal 1

```
total=0
suc=open('vtas-suc1.txt')
... # línea faltante
suc.close()
for elem in todas:
    lin=elem.strip('\n')
    lin=lin.split(',')
    total+=float(lin[1])
linea='Total ventas semestre sucursal 1 y 2: $'+str(total)
semestre=open('vtas-semestre.txt','w')
semestre.write(linea)
semestre.close()
```

A	`todas=suc.readlines()`	**X**	**A**
B	`suc.write(linea)`		**B**
C	`todas=open('vtas-suc1.txt', 'a')`		**C**
D	`suc=close('\n')`		**D**

Ejercicio 5 – Tema 2 | **1 Pto**

¿Cuál programa logra rellenar todos los elementos de la lista ***suma*** sin que se produzca fallo cuando se acaban los elementos de la lista ***lis1***? La salida correcta debe ser:
`[11, 22, -2, 2, 1, 1]`

Nota:
El operador `*` aplicado a una lista es el operador repetición
Ej:
[1,2]*3 -> [1,2,1,2,1,2]

A

```
lis1=[1,25,-8]
lis2=[10,-3,6,2,1,1]
suma=[]
for i in range(len(lis2)):
    total=lis1[i]+lis2[i]
    try:
        suma.append(total)
    except IndexError:
        print('No hay más lugar')
print(suma)
```

A

B

```
lis1=[1,25,-8]
lis2=[10,-3,6,2,1,1]
suma=[0]*len(lis1)
for i in range(len(lis2)):
    total=lis1[i]+lis2[i]
    try:
        suma[i]=total
    except IndexError:
        suma[i]=0
print(suma)
```

B

C

```
lis1=[1,25,-8]
lis2=[10,-3,6,2,1,1]
suma=[0]*len(lis2)
for i in range(len(lis2)):
    try:
        suma[i]=lis1[i]+lis2[i]
    except IndexError:
        suma[i]=lis2[i]
print(suma)
```

X

C

D

```
lis1=[1,25,-8]
lis2=[10,-3,6,2,1,1]
try:
    suma=[0]*len(lis2)
except ValueError:
    suma=lis1+lis2
for i in range(len(lis2)):
    suma[i]=lis1[i]+lis2[i]
print(suma)
```

D

Ejercicio 6 – Tema 2 | **2 Ptos**

Dado el Dataframe ***clientes***:

```
  apellido  edad       ciudad   provincia
0    Antón  50.0      Chajarí  Entre Ríos
1  Khunter   NaN    Concordia  Entre Ríos
2  Ortúzar  44.0   La Carlota     Córdoba
3    Gómez  23.0      Chajarí  Entre Ríos
4     Arce  35.0       La Paz  Entre Ríos
5    Alves  31.0  Salsipuedes     Córdoba
6    Alves  41.0         None        None
7     Díaz  72.0   Tafí Viejo     Tucumán
```

¿Qué salida produce la siguiente instrucción?

```
print(clientes[~(clientes['provincia'].isnull())&
               (clientes['edad']<50)])
```

A

```
    provincia       ciudad apellido
0  Entre Ríos      Chajarí    Antón
1  Entre Ríos    Concordia  Khunter
2     Córdoba   La Carlota  Ortúzar
3  Entre Ríos      Chajarí    Gómez
4  Entre Ríos       La Paz     Arce
5     Córdoba  Salsipuedes    Alves
6        None         None    Alves
7     Tucumán   Tafí Viejo     Díaz
```

A

B

```
  apellido  edad       ciudad   provincia
5    Alves  31.0  Salsipuedes     Córdoba
2  Ortúzar  44.0   La Carlota     Córdoba
4     Arce  35.0       La Paz  Entre Ríos
1  Khunter   NaN    Concordia  Entre Ríos
0    Antón  50.0      Chajarí  Entre Ríos
3    Gómez  23.0      Chajarí  Entre Ríos
7     Díaz  72.0   Tafí Viejo     Tucumán
```

B

C

```
  apellido  edad       ciudad provincia
5    Alves  31.0  Salsipuedes   Córdoba
6    Alves  41.0         None      None
7     Díaz  72.0   Tafí Viejo   Tucumán
```

C

D

```
  apellido  edad       ciudad   provincia
2  Ortúzar  44.0   La Carlota     Córdoba
3    Gómez  23.0      Chajarí  Entre Ríos
4     Arce  35.0       La Paz  Entre Ríos
5    Alves  31.0  Salsipuedes     Córdoba
```

X

D

Ejercicio 7 – Tema 2 | **2 Ptos**

Dado el siguiente programa:

```
def naipe1(t):
    return t

def naipe2(tup):
    return tup[0]+' de '+tup[1].upper()

def naipe3(tup):
    return list(tup)

def naipe4(tup):
    return tup[1]+', '+tup[0].upper()

mano=[('10','trébol'),('2','diamante'),
      ('10','diamante'),('Jota','trébol')]
.....   # línea de código  faltante
for carta in detalle:
    print(carta)
```

¿Cuál debería ser la línea de código faltante para que la salida sea la siguiente?

```
trébol, 10
diamante, 2
diamante, 10
trébol, JOTA
```

A	`detalle=list(map(naipe2,mano))`		**A**
B	`detalle.insert(0,edita3(mano))`		**B**
C	`detalle=list(map(naipe4,mano))`	**X**	**C**
D	`detalle=str(map(mano,naipe1))`		**D**

Ejercicio 8 – Tema 2 | **2 Ptos**

Dado el siguiente programa:

```
presentaciones=...  #??
pedido=[1,9,11,7]
print('Códigos de Artículos Pedidos')
print(*pedido)
print('Infusiones pedidas que tienen presentación en saquito:')
for elem in pedido:
    if elem in presentaciones and 'saquito' in
presentaciones[elem]:
        print(presentaciones[elem][0])
```

¿Cuál de las opciones de la estructura **presentaciones** funcionará correctamente para producir la siguiente salida?

```
Códigos de Artículos Pedidos
1 9 11 7
Infusiones pedidas que tienen presentación en saquito:
té
yerba
```

Nota:

Un * delante de una lista en un print() saca un elemento al lado del otro, separados por espacio

Ej: print(*[1,2,3]) -> 1 2 3

A	`presentaciones={'té':['hebras','saquito'], 'café':['granos','molido', 'cápsula','saquito'], 'yerba':['saquito','con palo', 'sin palo'], 'chocolatada':['tetra','polvo']}`		**A**
B	`presentaciones=['té', 'hebras', 'saquito', 'café', 'granos', 'molido', 'cápsula', 'saquito', 'yerba', 'saquito', 'con palo', 'sin palo', 'chocolatada', 'tetra', 'polvo']`		**B**
C	`presentaciones={1:['té','hebras','saquito'], 2:['café','granos','molido','cápsula', 'saquito'], 11:['yerba','saquito','con palo', 'sin palo'], 5:['chocolatada','tetra','polvo']}`	**X**	**C**
D	`presentaciones={'saquito':[ 'café', 'té', 'yerba'], 'molido':[ 'café'], 'granos':[ 'café'], 'polvo':[ 'chocolatada']}`		**D**

Ejercicio 9 – Tema 2 | **2 Ptos**

¿Cuál es la salida del siguiente programa?

```
def edita (lis):
    return lis[1]

def patron(lis):
    inicial=lis[0][0].lower()
    return inicial=='m'

vecinos=[['clara','díaz'],['MARTÍN','ARGAÑARAZ'],
         ['Silvio','Lusikcatz'],['mariela','ruiz'],
         ['maría','alves']]
datosValidos=list(filter(patron,vecinos))
salida=list(map(edita,datosValidos))
print(salida)
```

A	`['MARTÍN', 'MARÍA']`		**A**
B	`['ARGAÑARAZ', 'ruiz', 'alves']`	**X**	**B**
C	`[]`		**C**
D	`['díaz', 'lusikcatz', 'ruiz']`		**D**