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 0102 - 1 punto

¿Qué contiene la variable *b* ?

```
def organiza(n):
  resp=300-n
  return abs(resp)

a=[1,15,150,1500,2,20,200,2000]
b=list(map(organiza,a))
```

1.	**[299, 285, 150, -1200, 298, 280, 100, -1700]**		**1**
2.	**[]**		**2**
3.	**[299, 285, 150, 1200, 298, 280, 100, 1700]**	**X**	**3**
4.	**[300, 300, 300, -1200, 300, 300, 300, -1700]**		**4**

Ejercicio 0202 - 1 punto

¿Qué versión de la función ingreso() valida correctamente los datos de acceso de usuarios? Deben coincidir usuario y clave

```
def ingreso(usr):
   # desarrollo de la funcion
   -
   -

# users contiene usuario:clave
users={'jmiguel':'X25fc230','irojas':'Irma1211',
       'jsal':'joacoS01','lledesma':'LEDElun9'}
while not ingreso(users):
  print('Datos de Acceso Erróneos')
```

1.

```
def ingreso(usr):
  u=input('Usuario: ')
  cl=input('Clave: ')
  if u in usr and usr[u]==cl:
    return False
  return True
```

1

2.

```
def ingreso(usr):
  u=input('Usuario: ').upper()
  cl=input('Clave: ').upper()
  if cl==usr[u]:
    return False
```

2

	`  return True`		

3.	`def ingreso(usr):`
	`  u=input('Usuario: ').lower()`
	`  cl=input('Clave: ').lower()`
	`  if u in usr or cl==usr[u]:`
	`  return True`
			3

4.	`def ingreso(usr):`
	`  u=input('Usuario: ')`
	`  cl=input('Clave: ')`
	`  if u in usr and usr[u]==cl:`
	`  return True`
	`  else:`
	`  return False`
		X	**4**

Ejercicio 0302 - 1 punto

¿ Qué versión de la función entPositivo() valida correctamente el ingreso de un número natural (entero positivo)?
Nota: Se espera que la función no termine hasta obtener y devolver un valor correcto

```
def entPositivo(cartel):
  -
  -

num=entPositivo('Ingresá un número natural: ')
```

1	`def entPositivo(cartel):`
	`  para=False`
	`  while para:`
	`  try:`
	`  n=int(input(cartel))`
	`  para=True`
	`  except ValueError:`
	`  print('Número entero')`
	`  return n`
			1

2	`def entPositivo(cartel):`
	`  para=False`
	`  while not para:`
	`  try:`
	`  n=int(input(cartel))`
	`  if n>=1:`
	`  para=True`
	`  else:`
	`  print('Tiene que ser positivo')`
	`  except ValueError:`
	`  print('Número entero')`
	`  return n`
		X	**2**

3	`def entPositivo(cartel):`
	`  para=False`
	`  while para==False:`
	`  try:`
	`  n=int(input(cartel))`
	`  if n<=0:`
	`  print('Tiene que ser positivo')`
	`  para=True`
	`  except ValueError:`
	`  print('Número entero')`
	`  para=True`
	`  return n`
			3

4	`def entPositivo(cartel):`
	`  try:`
	`  n=int(input(cartel))`
	`  return n`
	`  except ValueError:`
	`  print('Número entero')`
			4

Ejercicio 0402 - 1 punto

¿Qué versión de la función cuenta() devuelve la cantidad de nombres que comienzan con consonantes en el archivo nombres, si este archivo tiene 3 líneas con nombres separados por -?

Contenido de nombres.txt:
Andrea-LAURA-MÁXIMO-Ileana
jorgelina-Silvana-amalia-alberto-andrés
ALMA-ELENA

cuenta() debe devolver 4

```
def cuenta(arch):
  -
  -

datos=open('nombres.txt',encoding='utf-8')
cantidad=cuenta(datos)
datos.close()
print(cantidad,'nombres inician con consonantes')
```

1

```
def cuenta(arch):
  voc='aeiouáéíóú'
  cant=0
  lineas=arch.readlines()
  for i in range(len(lineas)):
    if lineas[i][0].lower()not in voc:
        cant+=1
  return cant
```

1

2

```
def cuenta(arch):
  voc='aeiouáéíóú'
  lineas=arch.readlines()
  for i in range(len(lineas)):
    cant=0
    lin=lineas[i].split('-')
    for nom in lin:
      if nom.lower() in voc:
        cant+=1
  return cant
```

2

3

```
def cuenta(arch):
  voc='aeiouáéíóú'
  cant=0
  linea=arch.readline()
  linea=linea.split()
  for nom in linea:
    if nom[0].lower() in voc:
      cant+=1
  return cant
```

3

4

```
def cuenta(arch):
  voc='aeiouáéíóú'
  cant=0
  lineas=arch.readlines()
  for i in range(len(lineas)):
    lineas[i]=lineas[i].strip('\n').split('-')
    for nom in lineas[i]:
      if nom[0].lower() not in voc:
        cant+=1
  return cant
```

X **4**

Ejercicio 0502 - 1 punto

Dado el siguiente DataFrame *recaudacion*:

	cajero	caja	total
0	juan	3	104000.77
1	ana	1	256000.00
2	mario	5	99025.40
3	juan	1	112000.00
4	ana	2	77090.00
5	mario	3	86450.00

Que contiene 6 filas y 3 columnas: nombre del cajero (cajero), número de la caja en que cobró (caja) y total cobrado en esa ocasión (total).

¿Qué instrucción produce la siguiente salida?

```
      caja
1    256000.00
2     77090.00
3    104000.77
5     99025.40
```

1	`recaudacion['caja'].value_counts()`		**1**
2	`recaudacion.head(4)`		**2**
3	`recaudacion.loc[:,['total']]`		**3**
4	`recaudacion.groupby(recaudacion['caja'])['total'].max()`	**X**	**4**

Ejercicio 0602 - 1 punto

Dado el siguiente programa:

```
def obtieneNom(t):
  return t[:5].upper()

nombres=['ana López','emiliano SAL','LORENA ana báunes',
         'analía Soto','ángelea FALCÓN',
         'Luciana analía pérez','Ana María Giménez']
resultado= . . .
for nom in resultado:
  print(nom)
```

Qué produce la siguiente salida:

```
ANA L
EMILI
LOREN
ANALÍ
ÁNGEL
LUCIA
ANA M
>>>
```

Qué instrucción debería ir en los puntos suspensivos?

<u>Nota:</u> El argumento *key* 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	`list(map(obtieneNom,nombres))`	**X**	**1**
2	`list(filter(obtieneNom,nombres))`		**2**
3	`sorted(nombres,key=obtieneNom)`		**3**
4	`min(nombres,key=obtieneNom)`		**4**

Ejercicio 0702 - 1 punto

Dado el siguiente programa:

```
print('Ingresá números enteros, 0 para terminar')
num=1
i=1
lista=[]
while num!=0:
  num=input(str(i)+': ')
  try:
    num=int(num)
    lista.append(num)
  except ValueError:
    lista.append(0)
  i+=1
while 0 in lista:
  lista.remove(0)
```

Y los siguientes ingresos:

```
1: -55
2: -3.55
3: ese no era
4: 55*
5: 0
```

¿Qué contenido tendrá *lista* al finalizar?

1	`[-55, 55]`		**1**
2	`[]`		**2**
3	`[-55, -3.55, 0, 55, 0]`		**3**
4	`[-55]`	**X**	**4**

Ejercicio 0802 - 1 punto

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

1.	`ax.set_xlabel('Tiempo (días)')`		**1**
2.	`ax.plot(y, x, label='Ventas')`	**x**	**2**
3.	`ax.set_xlim(0, 6)`		**3**
4.	`ax.set_title("Gráfico de ventas")`		**4**

Ejercicio 0902 - 1 punto

Luego de ejecutar el siguiente fragmento de código:

```
x = [0, 1, 2, 3, 4, 5]
x_cubica = [0, 1, 8, 27, 64, 125]
x_bar = [2,3,1,6,2,1]

fig, ax = plt.subplots(nrows=4, ncols=5)
ax[0, 4].plot(x, x_cubica)
ax[2, 3].bar(x, x_bar)
plt.show()
```

¿Cuál de las siguientes opciones describe correctamente el gráfico que se ilustrara?

1	**Se hará una grilla de 20 gráficos en donde habrá dos gráficos de línea, uno en la primera fila y cuarta columna y el otro en la tercera fila y cuarta columna.**		**1**
2	**Se hará una grilla de 20 gráficos en donde habrá un gráfico de línea y uno de barras. El de línea en la primera fila y quinta columna, mientras que el de barras en la tercera fila y cuarta columna.**	X	**2**
3	**Se hará una grilla de 20 gráficos en donde habrá un gráfico de línea y uno de barras. El de línea en la primera fila y cuarta columna, mientras que el de barras en la segunda fila y tercera columna.**		**3**
4	**Ninguna de las opciones describe lo que ocurre al ejecutar el código**		**4**

Ejercicio 1002 - 2 puntos

¿Cómo queda el archivo vtasSemestre.txt luego de ejecutar el siguiente programa?

```
def leer(arch):
  vtasMes={1:0,2:0,3:0,4:0,5:0,6:0}
  ventas=open(arch,encoding='utf-8')
  detalle=ventas.readlines()
  ventas.close()
  for i in range(len(detalle)):
    detalle[i]=detalle[i].strip().split(',')
    mes=int(detalle[i][0])
    vta=float(detalle[i][1])
    vtasMes[mes]=vta
  return vtasMes

def guarda(dicci,arch,meses):
  datos=open(arch,'w',encoding='utf-8')
  for mes in dicci:
    lin=meses[mes].upper()+':'+str(dicci[mes])+','
    datos.write(lin)
  datos.close()

meses= '','enero','febrero','marzo','abril','mayo','junio'
ventas=leer('detalleVtas.txt')
guarda(ventas,'ultVta.txt',meses)
```

Nota: El contenido de detalleVtas.txt es el siguiente:
1,25
2,33
1,33.5
3,20
4,10
1,11

Por cada línea viene el número de mes y el valor de una venta separados por coma

1	**1:11** **2:33.0** **3:20.0** **4:10.0**		**1**
2	**enero:11.0** **febrero:33.0** **marzo:20.0** **abril:10.0** **mayo:0** **junio:0**		**2**
3	**ENERO:11.0,FEBRERO:33.0,MARZO:20.0,ABRIL:10.0,MAYO:0,JUNIO:0,**	**X**	**3**
4	**mayo,junio**		**4**

Ejercicio 1102 - 2 puntos

Dado el siguiente DataFrame *menu*:

	nombre	principal	calorías
lunes	tallarines	hidratos	1500
martes	milanesas	proteínas	1800
miércoles	tarta de choclo	hidratos	1000
jueves	lenguado	proteínas	950

Que contiene 4 filas (el día de la semana es el índice de la fila) y 3 columnas: nombre del plato (nombre), grupo nutricional principal (principal) y calorías por porción (calorías).

¿Qué muestra como resultado la siguiente instrucción?

```
menu.loc[:,['principal','nombre']]
```

1.

	nombre	principal	calorías
Lunes	tallarines	hidratos	1500

2. X

	principal	nombre
Lunes	hidratos	tallarines
Martes	proteínas	milanesas
Miércoles	hidratos	tarta de choclo
Jueves	proteínas	lenguado

3.

	calorías	principal	nombre
Lunes	1500	hidratos	tallarines

4.

	calorías	principal	nombre
Lunes	1500	hidratos	tallarines
Martes	1800	proteínas	milanesas
Miércoles	1000	hidratos	tarta de choclo
Jueves	950	proteínas	lenguado