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 10 preguntas de opción múltiple. Cada pregunta tiene una y sólo una respuesta correcta.

Las respuestas deben completarse 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**
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

Dado el siguiente condicional:

```
if c in (44,0,-5):
  print('Luna de agua')
elif (a%3==0 and b<51) or (a%3==1) or (b>51 and a==0):
  print('Noche estrellada')
else:
  print('Noche tormentosa')
```

Qué valores de ***a***,***b*** y ***c*** permiten que se imprima: **Noche tormentosa** ?

1	`a=33, b=50, c=100`		**1**
2	`a=5, b=52, c=10`	**X**	**2**
3	`a=30, b=50, c=0`		**3**
4	`a=10, b=50, c=100`		**4**

Ejercicio 0202 - 1 punto

¿Qué muestra por pantalla el siguiente programa?

```
j=0
for i in range(3):
  j+=10
while i<5:
  j-=2
  m=j
  for k in range(8,m//2):
    j-=1
  i+=1
texto='$'*j
print(texto)
```

<u>Nota</u>: El operador * aplicado a una string repite n veces la misma
Ej:
`print('a'*3)` **muestra** `aaa`

1	`$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$`		**1**
2	`$` `$`		**2**
3	`'$'*j`		**3**

4	`$$$$$$$$$$$$$$$`	**X**	**4**

Ejercicio 0302 - 1 punto

¿Con qué contenido acabará el diccionario *contLetras* cuando se ejecute el siguiente programa?

```
ciudades=['añatuya','CERES','Paraná']
contLetras={}
for nom in ciudades:
  for let in nom.lower():
    if let not in contLetras:
      contLetras[let]=1
    else:
      contLetras[let]+=1
```

1	`{'a': 5, 'ñ': 1, 't': 1, 'u': 1, 'y': 1, 'c': 1, 'e': 2, 'r': 2, 's': 1, 'p': 1, 'n': 1, 'á': 1}`	**X**	**1**
2	`{}`		**2**
3	`{'AÑTUYCERSPNÁ': 18}`		**3**
4	`{5: 'A',1: 'Ñ',1: 'T',1: 'U',1: 'Y',1: 'C',2: 'E',2: 'R', 1: 'S',1: 'P',1: 'N',1: 'Á'}`		**4**

Ejercicio 0402 - 1 punto

Si *lis*=[1,22,4,1,6,5,1,10], cuál código NO logra traer todos los 1s al principio de la lista? *lis* debería quedar [1, 1, 1, 22, 4, 6, 5, 10] Es decir, del mismo tamaño pero con todos los 1s que tenga al inicio, y los elementos que no sean 1 al fondo en el mismo orden original.

<u>Notas</u>:
El operador * aplicado a una lista repite n veces la misma
Ej:
`[0]*3` **genera** `[0,0,0]`
El método *extend*() agrega el contenido de una lista al final de otra
Ej:
`a=[1,0]`
`a.extend([2,2])` **deja la lista** `a` **en** `[1,0,2,2]`

1	`cont=0`
	`while 1 in lis:`
	`  lis.remove(1)`
	`  cont+=1`
	`inicio=[1]*cont`
	`inicio.extend(lis)`
	`lis=inicio`
			1

2	`cont=lis.count(1)`
	`while 1 in lis:`
	`  lis.remove(1)`
	`lis=([1]*cont)+lis`
			2

3	`unos=[]`
	`otros=[]`
	`for n in lis:`
	`  if n==1:`
	`  unos.append(n)`
	`  else:`
	`  otros.append(n)`
	`lis=unos+otros`
			3

4	`for i in range(len(lis)):`
	`  n=lis.pop(i)`
	`  if n==1:`
	`  lis.insert(i,n)`
	`  else:`
	`  lis.append(n)`
		X	**4**

Ejercicio 0502 - 1 punto

¿Qué versión de la función *natural* funcionará correctamente en el siguiente programa para obtener un ingreso por teclado de un número natural (entero y mayor que 0)? La función debe insistir hasta que el número ingresado sea efectivamente un natural.

```
def natural(…):
  -
  -
  -

a=natural()
b=natural()
print(a+b)
```

1

```
def natural():
  pide=False
  while pide:
    try:
      num=int(input('Número natural: '))
      if num>0:
        pide=False
      else:
        pide=True
        print('Ingrese un número natural')
    except ValueError:
      print('Ingrese un número natural')
  return num
```

2

```
def natural(num):
  while True:
    try:
      num=int(input('Número natural: '))
      if num<=0:
        print('Ingrese un número natural')
    except ValueError:
      print('Ingrese un número natural')
  return num
```

3 **X**

```
def natural():
  while True:
    try:
      num=int(input('Número natural: '))
      if num>0:
        return num
      else:
        print('Ingrese un número natural')
    except ValueError:
      print('Ingrese un número natural')
```

4

```
def natural():
  pide=True
  while pide:
    try:
      num=int(input('Número natural: '))
      pide=False
      if num<=0:
        print('Ingrese un número natural')
    except ValueError:
      print('Ingrese un número natural')
      pide=False
  return num
```

Ejercicio 0602 - 1 punto

¿Qué muestra el siguiente programa?

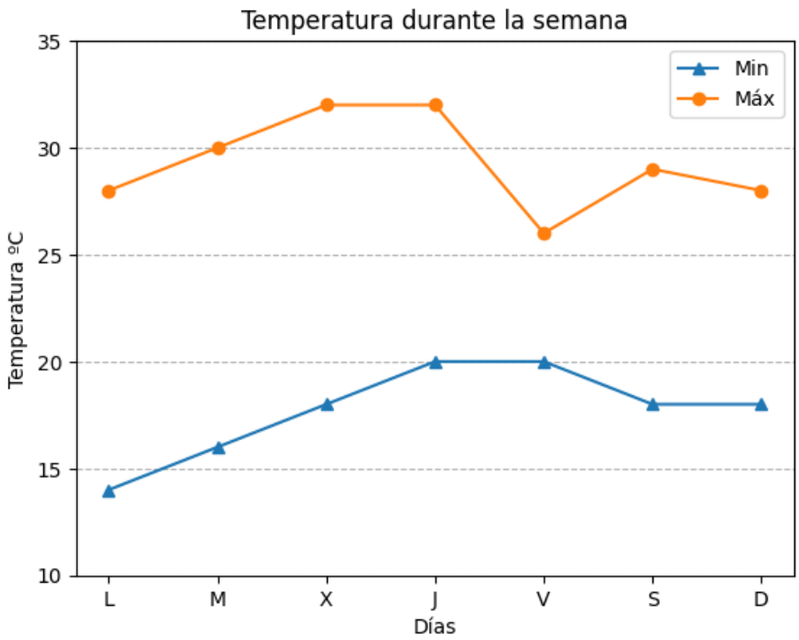
```
def letras(num):
  let='AEIOU'
  return let[abs(num)%5]

lista=[2,12,0,10,-3]
nueva=list(map(letras,lista))
print(nueva)
```

1	**['i', 'a', 'o']**		**1**
2	**['I', 'I', 'A', 'A', 'O']**	**X**	**2**

3	['o', 'a', 'a', 'i', 'i']		3
4	['AEIOU']		4

Ejercicio 0702 - 1 punto



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

1	`ax.scatter(dias, temperaturas_min, marker = '^', label = 'Min')` `ax.scatter(dias, temperaturas_max, marker = 'o', label = 'Máx')`	X	1
2	`ax.grid(axis = 'y', linestyle = 'dashed')`		2
3	`ax.set_title("Temperatura durante la semana")`		3
4	`ax.set_xlabel("Días")`		4

Ejercicio 0802 - 2 puntos

¿Qué versión de *elige* funcionará correctamente en este programa?

```
def elige(…):
  -
  -
  -

gustos={'f':'Frutilla','v':'Vainilla',
        'd':'Dulce de leche','c':'Chocolate'}
salsas={'d':'Ddl','c':'Chocolate','f':'Frutilla'}
copa=[]
print('Armá tu copa, 3 sabores')
for i in range(1,4):
  copa.append(elige('Sabor '+str(i),gustos))
copa.append(elige('Qué salsa?',salsas))
print('Mi copa es de',copa[0],',',copa[1],'y',copa[2],
                'con salsa de',copa[3])
```

1	`def elige(opciones):`
	`  print('Elegí tu opción con la letra resaltada')`
	`  print('pregunta')`
	`  for i in range(len(opciones)):`
	`  print(opciones[i])`
	`  sel=input().lower()`
	`  while sel not in opciones:`
	`  sel=input().lower()`
	`  return sel`
			1

2	`def elige(opciones,pregunta):`
	`  print('Elegí tu opción con la letra resaltada')`
	`  print(pregunta)`
	`  for opc in opciones:`
	`  print(opciones[opc])`
	`  sel=input()`
	`  while sel in opciones:`
	`  sel=input()`
	`  return opciones[sel]`
			2

3	def elige(pregunta,opciones):
	  print('Elegí tu opción con la letra resaltada')
	  print(pregunta)
	  for opc in opciones:
	  print(opciones[opc])
	  sel=input().lower()
	  while sel not in opciones:
	  sel=input().lower()
	  return opciones[sel]
		X	3

4	def elige(opciones):
	  print('Elegí tu opción con la letra resaltada')
	  print('pregunta')
	  for i in range(len(opciones)):
	  print(opciones[i])
	  sel=input().lower()
	  while sel not in range(len(opciones)):
	  sel=input().lower()
	  return sel
			4

Ejercicio 0902 - 2 puntos

¿ Qué valor para el argumento *modo* debería pasársele a la función *abrir* (en la primera y en la segunda invocación) para permitir que el programa guarde en mayúsculas las palabras del archivo *ar.txt*?

```
def abrir(arch,modo):
  archivo=open(arch,modo,encoding='utf-8')
  return archivo

verbos=abrir('ar.txt',...) #Primera
lineas=verbos.readlines()
verbos.close()
verbos=abrir('ar.txt',...) #Segunda
for lin in lineas:
  verbos.write(lin.upper())
verbos.close()
```

1	**Primera:** `'r'` **Segunda:** `'a'`	X	1
2	**Primera:** `'r+'` **Segunda:** `'w'`	X	2
3	**Primera:** `'w+'` **Segunda:** `'w'`		3
4	**Primera:** `'a'` **Segunda:** `'r+'`		4

Dado que el enunciado es ambiguo y se puede interpretar como agregar o reemplazar, las opciones marcadas son las correctas.

Ejercicio 1002 - 2 puntos

Para el siguiente DataFrame *arts*:

```
         Cgo  Desc                      Pr Unit   Exist
0     LIB001  Lápiz negro HB              450.0    55.0
1     PAP112  Res Ledesma 70gr AA        2700.0    NaN
2     PAP099  Blck cuad Rivadavia A4       958.5    2.0
3     MAP070  Planisferio nro 5            210.0    13.0
4     LIB015  Marc negro Sylvapen          675.0    11.0
5     LIB118  Regl Pizzini 30cm            387.0    NaN
```

¿Qué instrucción produce la siguiente salida?

```
     Pr Unit Desc                   Cgo
0      450.0 Lápiz negro HB         LIB001
1     2700.0 Res Ledesma 70gr AA    PAP112
2      958.5 Blck cuad Rivadavia A4 PAP099
3      210.0 Planisferio nro 5      MAP070
4      675.0 Marc negro Sylvapen    LIB015
5      387.0 Regl Pizzini 30cm      LIB118
```

1	`arts[arts['Pr Unit'].between(300,2700)]`		1

2	`arts.head()`		**2**
3	`arts[['Pr Unit','Desc','Cgo']]`	**X**	**3**
4	`arts[(arts['Pr Unit']<900)&(arts['Exist']>10)]`		**4**