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 0104 - 1 punto

Dado el siguiente condicional:

```
if a in c:
  print('De una')
elif (a.upper()>'E' and b) or (a<'m') or (not b and a=='*'):
  print('Sale segunda')
elif a>'c' and a<'u':
  print('Última opción')
```

Qué valores de ***a***,***b*** y ***c*** permiten que se imprima: **Sale segunda** ?

1	`a='E', b=False, c='DALE'`		**1**
2	`a='o', b=False, c='dale'`		**2**
3	`a='z', b=False, c='una'`		**3**
4	`a='m', b=True, c='dale'`	**X**	**4**

Ejercicio 0204 - 1 punto

¿Qué muestra por pantalla el siguiente programa?

```
j=5
for i in range(3):
  j+=15
while i<5:
  for k in range(10,15-i):
    j-=5
  i+=1
print(j)
```

1	`20`	**X**	**1**
2	`0`		**2**
3	`-111`		**3**
4	`100`		**4**

Ejercicio 0304 - 1 punto

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

```
parrafo='desayuno PORTEÑO incluye café con Leche y 2 medialunas'
contCons={}
```

```
for let in parrafo:
  l=let.upper()
  if l.isalpha() and l not in 'ÁÉÍÓÚAEIOU':
    if l not in contCons:
      contCons[l]=0
    contCons[l]+=1
```

Nota: El método *isalpha()* devuelve True si el argumento es una letra, False en caso contrario

1	`{}`		**1**
2	`{'e': 6, 'a': 4, 'u': 3, 'o': 4, 'i': 2}`		**2**
3	`{35: 'dsyn prtñclféh2m'}`		**3**
4	`{'D': 2, 'S': 2, 'Y': 3, 'N': 4, 'P': 1, 'R': 1, 'T': 1, 'Ñ': 1, 'C': 4, 'L': 3, 'F': 1, 'H': 1, 'M': 1}`	**X**	**4**

Ejercicio 0404 - 1 punto

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

Nota:
El operador * aplicado a una lista repite n veces la misma
Ej:
`[0]*3` **genera** `[0,0,0]`

1

```
def par(n):
  return n%2==0

lis=…
pares=list(filter(par,lis))
pares.reverse()
impares=pares
for n in lis:
  if n%2!=0:
    impares.append(n)
lis=impares
```

X **1**

2

```
pares=[]
impares=[]
for n in lis:
  if n%2==0:
    pares.append(n)
  else:
    impares.append(n)
lis=impares+pares
```

2

3

```
def par(n):
  return n%2==0

lis=…
pares=list(filter(par,lis))
pares=[0]*(len(lis)-len(pares))+pares
j=0
for i in range(len(lis)):
  if lis[i]%2!=0:
    pares[j]=lis[i]
    j+=1
lis=pares
```

3

4

```
def par(n):
  return n%2==0

def impar(n):
  return n%2==1

lis=…
pares=list(filter(par,lis))
impares=list(filter(impar,lis))
lis=impares+pares
```

4

Ejercicio 0504 - 1 punto

¿Qué versión de la función *negativo* funcionará correctamente en el siguiente programa para obtener un ingreso por teclado de un número negativo? La función debe insistir hasta que el número ingresado sea efectivamente un valor negativo.

```
def negativo(…):
  -
  -
  -

a=negativo()
b=negativo()
print(a-b)
```

1

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

X **1**

2

```
def negativo():
  try:
    num=input('Número negativo: ')
    num=float(num)
    if num<0:
      return num
    return 0
  except ValueError:
    print('Ingrese un número negativo')
```

2

3

```
def negativo():
  dale=False
  while dale:
    try:
      num=input('Número negativo: ')
      num=float(num)
      if num<0:
        dale=False
      else:
        print('Ingrese un número negativo')
        dale=True
    except ValueError:
      print('Ingrese un número negativo')
```

3

4

```
def negativo():
  dale=True
  while dale:
    try:
      num=input('Número negativo: ')
      num=float(num)
      if num>=0:
        dale=True
      else:
        print('Ingrese un número negativo')
        dale=False
    except ValueError:
      print('Ingrese un número negativo')
```

4

Ejercicio 0604 - 1 punto

¿Qué muestra el siguiente programa?

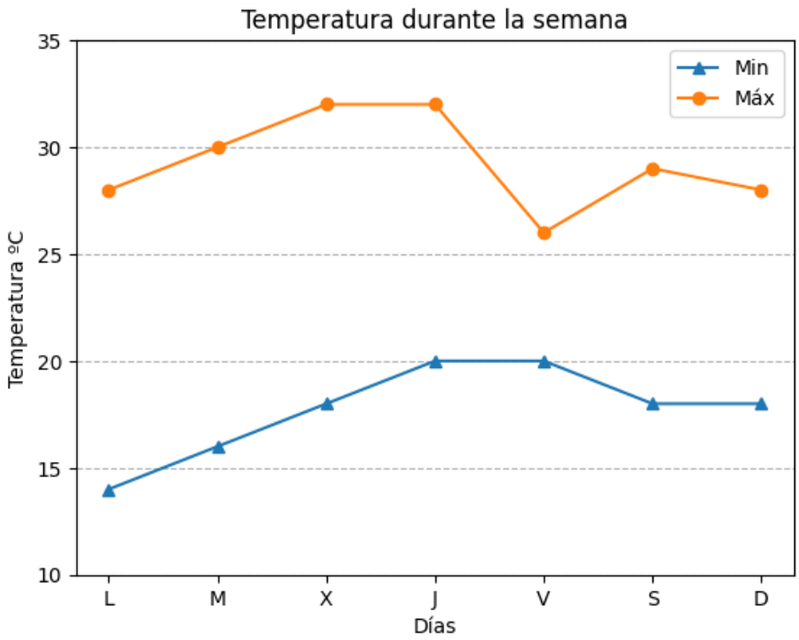
```
def letras(num):
  let='ToTaLeS'
  return let[abs(num)%7]

lista=[-1,2,16,-7,0]
nueva=list(map(letras,lista))
print(nueva)
```

1 **['o', 't']** **1**

2	['o', 'T', 'T', 'T', 'T']	X	2
3	['ToTaLeS']		3
4	[]		4

Ejercicio 0704 - 1 punto



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

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

Ejercicio 0804 - 2 puntos

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

```
def elige(…):
  -
  -
  -

salsas={'f':'Filetto','b':'Boloñesa','c':'Crema','p':'Pesto',
        's':'Sin salsa'}
pastas={'f':'Fideos cinta','r':'Ravioles de calabaza',
        'm':'fideos Moñito'}
plato=[]
print('Armá tu plato de Pasta, hasta 2 salsas')
plato.append(elige(pastas,'Qué pasta?'))
for i in range(1,3):
  plato.append(elige(salsas,'Salsa '+str(i)))
print('Mi plato de pastas es',plato[0],'con',plato[1],'y',plato[2])
```

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

2	`def elige(opciones,pregunta):`
	`  print('Elegí tu opción con la letra resaltada')`
	`  print(pregunta)`
	`  for i in range(len(opciones)):`
	`  print(opciones[i])`
	`  sel=input().lower()`
			2

	`while sel not in opciones:`
	`  sel=input().lower()`
	`return sel`		

3	`def elige(opciones,pregunta):`
	`  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 opc in opciones:`
	`  print(opc,opc)`
	`  sel=input()`
	`  if sel in opciones:`
	`  return 'f'`
	`  else:`
	`  return opciones[sel]`
			4

Ejercicio 0904 - 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:** `'w'`	**X**	**1**
2	**Primera:** `'r+'` **Segunda:** `'a'`	**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 1004 - 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 A4958.5    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?

```
      Cgo         Desc                    Pr Unit Exist
0     LIB001      Lápiz negro HB          450.0   55.0
3     MAP070      Planisferio nro 5       210.0   13.0
4     LIB015      Marc negro Sylvapen     675.0   11.0
```

1	`arts.head()`		**1**

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