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 0105 - 1 punto

¿Qué contiene *b* ?

```
def organiza(n):
  res=False
  if n>=100 and n<1000:
    res=True
  return res

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

1	**[1,15,150,1500]**		**1**
2	**[1500, 2000]**		**2**
3	**[]**		**3**
4	**[150, 200]**	**X**	**4**

Ejercicio 0205 - 1 punto

¿Cuál versión de la función ingreso() valida correctamente los datos de acceso a una cuenta de mail? Deben coincidir dirección de mail y password

```
def ingreso(usr):

def ingreso(usr):
   -
   -

# users contiene [password,dirección mail]
users=[['X25fc230','info@usina.com.ar'],
       ['Irma1211','iro@hotmail.com'],
       ['joacoS01','ergo@gmail.com']]
while ingreso(users):
  print('Datos de Acceso Erróneos')
```

1.

```
def ingreso(usr):
  dccMail=input('Mail: ')
  clave=input('Password: ')
  if [clave,dccMail] in usr:
    return True
  else:
    return False
```

1

2.

```
def ingreso(usr):
  dccMail=input('Mail: ').lower()
  clave=input('Password: ').lower()
  return usr.count([clave,dccMail])>0
```

2

3.	`def ingreso(usr):`
	`  dccMail=input('Mail: ').upper()`
	`  clave=input('Password: ').upper()`
	`  return usr.count([clave,dccMail])==0`
			3

4.	`def ingreso(usr):`
	`  dccMail=input('Mail: ')`
	`  clave=input('Password: ')`
	`  if not[clave,dccMail] in usr:`
	`  return True`
		X	**4**

Ejercicio 0305 - 1 punto

¿Cuál versión de la función intercala() devuelve una lista con los datos intercalados de dos listas que recibe, independiente de la longitud de cada una de ellas y el orden de intercalación?
Nota: intercala(a,b) debe devolver [1,10,2,20,3,30,40,50]
intercala(b,a) debe devolver [10,1,20,2,30,3,40,50]

```
def intercala(lis1,lis2):
  -
  -

a=[1,2,3]
b=[10,20,30,40,50]
c=intercala(a,b)
d=intercala(b,a)
```

1	`def intercala(lis1,lis2):`
	`  lisTot=[]`
	`  for i in range(len(lis2)):`
	`  lisTot.append(lis1[i])`
	`  try:`
	`  lisTot.append(lis2[i])`
	`  except IndexError:`
	`  nada=0`
	`  return lisTot`
			1

2	`def intercala(lis1,lis2):`
	`  lisTot=[]`
	`  for i in range(len(lis1)):`
	`  try:`
	`  lisTot.append(lis1[i])`
	`  except IndexError:`
	`  nada=0`
	`  lisTot.append(lis2[i])`
	`  return lisTot`
			2

3	`def intercala(lis1,lis2):`
	`  lisTot=[]`
	`  for i in range(len(lis1)+len(lis2)):`
	`  lisTot.append(lis1[i])`
	`  try:`
	`  lisTot.append(lis2[i])`
	`  except IndexError:`
	`  nada=0`
	`  return lisTot`
			3

4	`def intercala(lis1,lis2):`
	`  lisTot=[]`
	`  for i in range(len(lis1)):`
	`  lisTot.append(lis1[i])`
	`  try:`
	`  lisTot.append(lis2[i])`
	`  except IndexError:`
	`  nada=0`
	`  for i in range(len(lis1),len(lis2)):`
	`  lisTot.append(lis2[i])`
	`  return lisTot`
		X	**4**

Ejercicio 0405 - 1 punto

¿Cuál versión de la función guarda() genera un archivo personal.txt que contendrá únicamente los datos que le pasa el programa? El archivo debe quedar con el siguiente formato y contenido:
PAZ,ana
ALZA,inés
ORTIZ,sergio

```
def guarda(dicci,arch):
  -
  -

nombres={'ana':'paz','inés':'alza','sergio':'ortiz'}
guarda(nombres,'personal.txt')
```

1	`def guarda(dicci,arch):`
	`  datos=open(arch,'w',encoding='utf-8')`
	`  for nom in dicci:`
	`  datos.write(dicci[nom].upper()+','+nom+'\n')`
	`  datos.close()`
		X	**1**

2	`def guarda(dicci,arch):`
	`  datos=open(arch,'a',encoding='utf-8')`
	`  for nom in dicci:`
	`  datos.write(dicci[nom]+','+nom)`
	`  datos.close()`
			2

3	`def guarda(dicci,arch):`
	`  datos=open(arch,'w',encoding='utf-8')`
	`  datos.write(nom+','+dicci[nom]+'\n\n')`
	`  datos.close()`
			3

4	`def guarda(dicci,arch):`
	`  lineas=[]`
	`  for nom in dicci:`
	`  lineas.insert(0,'\n'+dicci[nom].lower()+' '+dicci[1]+'\n')`
	`  datos=open(arch,'w',encoding='utf-8')`
	`  datos.writelines(lineas)`
	`  datos.close()`
			4

Ejercicio 0505 - 1 punto

Dado el siguiente DataFrame *lluvias*:

	localidad	mes	mm
0	azul	abril	65
1	charata	abril	96
2	azul	julio	30
3	azul	octubre	72
4	federación	enero	110
5	quitilipi	marzo	120
6	federación	marzo	115

Que contiene 7 filas y 3 columnas: localidad, mes y precipitación total registrada en mm (mm).

¿Qué instrucción produce la siguiente salida?

```
      localidad  mes      mm
0     azul       abril    65
2     azul       julio    30
3     azul       octubre  72
1     charata    abril    96
4     federación enero    110
6     federación marzo    115
5     quitilipi  marzo    12
```

1	`lluvias.loc[2:6,['mm','localidad']]`		**1**
2	`lluvias.sort_values(by=['localidad','mes'], ascending=[True,True])`	**X**	**2**
3	`lluvias.iloc[2:6]`		**3**
4	`lluvias.groupby(lluvias['mes'])['localidad','mm'].max()`		**4**

Ejercicio 0605 - 1 punto

Dado el siguiente programa:

```
def obtieneNom(t):
  return 'ana' in t.lower()

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)
```

Que produce la siguiente salida:

ana López
LORENA ana báunes
analía Soto
Luciana analía pérez
Ana María Giménez
>>>
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))`		**1**
2	`list(filter(obtieneNom,nombres))`	**X**	**2**
3	`list(reversed(nombres))`		**3**
4	`list(sorted(nombres,key=obtieneNom))`		**4**

Ejercicio 0705 - 1 punto

Dado el siguiente programa:

```
print('Ingresá números, < para terminar')
sigue=True
i=1
numeros=[]
while sigue:
  num=input(str(i)+': ')
  try:
    num=float(num)
    numeros.insert(0,num)
  except ValueError:
    if num=='<':
      sigue=False
  i+=1
```

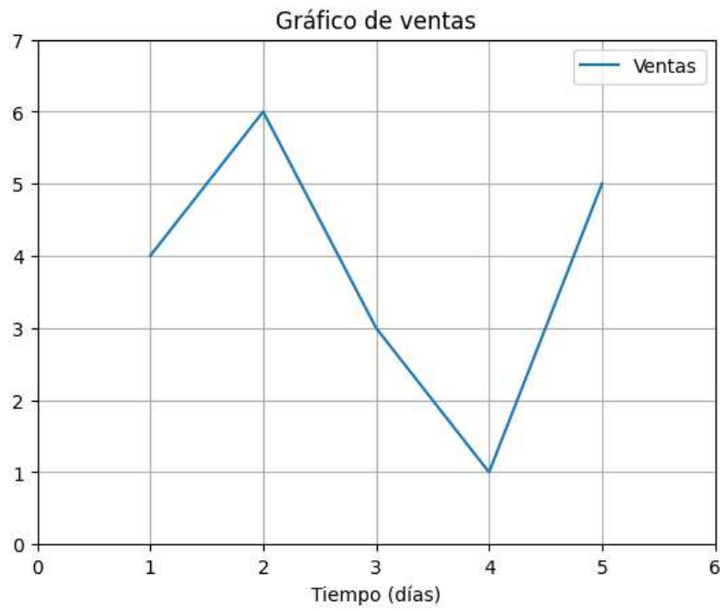
Y los siguientes ingresos:

```
1: 33
2: 4.66
3: -.88
4: 9*-
5: 0
6: <
```

¿Qué contenido tendrá *numeros* al finalizar?

1	`[33, 4, 0, 0, 0]`		**1**
2	`[33.0, 4.66, -0.88, 9, 0, '<']`		**2**
3	`[0.0, -0.88, 4.66, 33.0]`	**X**	**3**
4	`[]`		**4**

Ejercicio 0805 - 1 punto



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

1.	`ax.set_xlabel('Tiempo (días)')`		**1**
2.	`ax.set_ylabel('Tiempo (días)')`	**x**	**2**
3.	`ax.set_xlim(0, 6)`		**3**
4.	`ax.grid()`		**4**

Ejercicio 0905 - 1 punto

Se tiene el siguiente archivo llamado 'edades.txt':

```
Juan;20
Abril;22
Mario;21
```

¿Qué se va a imprimir luego de ejecutar el siguiente fragmento de código?

```
file = open('edades.txt', "r")
lines = file.readlines()
x = ''
for line in lines:
  x += line.split(';')[1]
print(x)
file.close()
```

1.	**63**		**1**
2.	**20** **22** **21**	**X**	**2**
3.	**JuanAbrilMario**		**3**
4.	**Se imprime algo diferente/Hay un error en el código y no se ejecuta**		**4**

Ejercicio 1005 - 2 puntos

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

```
def leer(arch):
  txtArch=open(arch,encoding='utf-8')
  txt=txtArch.read()
  txtArch.close()
  txtLis=txt.strip('\n').split()
  infinitivos=[]
  termina='ra','re','ri'
  for pal in txtLis:
    if pal[::-1][:2] in termina:
      infinitivos.append(pal)
  return infinitivos

def guarda(lista,arch):
  datos=open(arch,'w',encoding='utf-8')
  for pal in lista:
    datos.write(pal.upper()+'\n')
  datos.close()

palabras=leer('parrafo.txt')
guarda(palabras,'arErIr.txt')
```

Nota: El contenido de parrafo.txt es el siguiente:

la solución a convenir es mejor que pensar entender o ignorar la solución

1.	**CONVENIRPENSARENTENDERIGNORAR**		**1**
2.	**CONVENIR** **PENSAR** **ENTENDER** **IGNORAR**	**X**	**2**
3.	**la solución a es mejor que o la solución**		**3**
4.	**ignorar** **entender** **pensar** **mejor** **convenir**		**4**

Ejercicio 1105 - 2 puntos

Dado el siguiente DataFrame *cobertura*:

	región	provincia	sucursales
0	cuyo	san juan	3
1	litoral	santa fé	0
2	cuyo	mendoza	5
3	patagonia	chubut	2
4	cuyo	san luis	1
5	pampa	buenos aires	8

Que contiene 6 filas y 3 columnas: región geográfica (región), provincia y cantidad de sucursales abiertas (sucursales).

¿Qué se muestra como resultado al ejecutar las siguientes instrucciones?

```
cobertura['región']=cobertura['región'].map({'cuyo':'zona cuyana',
                                             'pampa':'pampa húmeda'})
cobertura[cobertura['región'].isnull()]
```

1	**región** 0 zona cuyana 1 NaN 2 zona cuyana 3 NaN 4 zona cuyana 5 pampa húmeda		**1**
2	**sucursales** 0		**2**
3	**región** **provincia** **sucursales** 1 NaN santa fé 0 3 NaN chubut 2	**X**	**3**
4	**región** **provincia** **sucursales** 1 litoral santa fé 0 3 patagonia chubut 2		**4**