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 0103 - 1 punto

¿Qué contiene *b* ?

```
def organiza(n):
  resp=n**3
  return abs(resp)

a=[1,-5,10,2,-20,6]
b=list(map(organiza,a))
```

1.	**[1, 125, 1000, 8, 8000, 216]**	**X**	**1**
2.	**[]**		**2**
3.	**[1, 1, 1, 1, 1, 1]**		**3**
4.	**[1, -125, 1000, 8, -8000, 216]**		**4**

Ejercicio 0203 - 1 punto

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

```
def ingreso(usr):

def ingreso(usr):
   -
   -

# users contiene [usuario,clave]
users=[['jmiguel','X25fc230'],
       ['irojas','Irma1211'],
       ['jsal','joacoS01']]

while not ingreso(users):
  print('Datos de Acceso Erróneos')
```

1.

```
def ingreso(usr):
  u=input('Usuario: ')
  cl=input('Clave: ')
  resp=False
  for elem in usr:
    if elem[0]==u and elem[1]==cl:
      resp=True
    else:
      resp=False
  return resp
```

1

2.	`def ingreso(usr):`
	`  u=input('Usuario: ')`
	`  cl=input('Clave: ')`
	`  return [u,cl] in usr`
		X	**2**

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

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

Ejercicio 0303 - 1 punto

¿Cuál versión de la función realInter() valida correctamente el ingreso de un número real en el intervalo [0,1] (números entre 0 y 1 inclusive)?
Nota: Se espera que la función no termine hasta obtener y devolver un valor correcto

```
def realInter(cartel,desde,hasta):
  -
  -

num=realInter('Ingresá un número real entre 0 y 1 inclusive: ',0,1)
```

1	`def realInter(cartel,desde,hasta):`
	`  try:`
	`  n=float(input(cartel))`
	`  if n>=desde or n<=hasta:`
	`  return n`
	`  else:`
	`  print('entre',desde,'y',hasta)`
			1

2	`def realInter(cartel,desde,hasta):`
	`  try:`
	`  n=float(input(cartel))`
	`  if n>=desde:`
	`  return n`
	`  else:`
	`  print('entre',desde,'y',hasta)`
	`  except ValueError:`
	`  print('Número real')`
			2

3	`def realInter(cartel,desde,hasta):`
	`  para=False`
	`  while para==False:`
	`  try:`
	`  n=float(input(cartel))`
	`  if n<=hasta and n>=desde:`
	`  para=1`
	`  else:`
	`  print('entre',desde,'y',hasta)`
	`  except ValueError:`
	`  print('Número real')`
	`  return n`
		X	**3**

4	`def realInter(cartel,desde,hasta):`
	`  para=True`
	`  while para==False:`
	`  try:`
	`  n=float(input(cartel))`
	`  if n<=hasta and n>=desde:`
	`  para=True`
	`  else:`
	`  print('entre',desde,'y',hasta)`
	`  except ValueError:`
	`  print('Número real')`
	`  para=True`
	`  return n`
			4

Ejercicio 0403 - 1 punto

¿Cuál versión de la función cuenta() devuelve la cantidad de nombres de longitud par en el archivo nombres, que 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 6

```
def cuenta(arch):
  -
  -

datos=open('nombres.txt',encoding='utf-8')
cantidad=cuenta(datos)
datos.close()
print(cantidad,'nombres tienen un largo par')
```

1

```
def cuenta(arch):
  lin=arch.readline()
  linea=lin.strip()
  cant=0
  for nom in linea:
    if len(nom)%2==1:
      cant+=1
  return cant
```

1

2

```
def cuenta(arch):
  lin=arch.readlines()
  cant=0
  for linea in lin:
    linea=linea.strip().split('-')
    for nom in linea:
      if len(nom)%2==0:
        cant+=1
  return cant
```

X **2**

3

```
def cuenta(arch):
  lin=arch.readlines()
  for linea in lin:
    linea=linea.strip().split('-')
    cant=0
    for nom in linea:
      if len(nom)==2:
        cant+=1
  return cant
```

3

4

```
def cuenta(arch):
  for i in range(3):
    linea=arch.readline()
    for nom in linea:
      if len(nom)%2!=0:
        cant+=1
  return cant
```

4

Ejercicio 0503 - 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?

```
      cajero     caja  total
1     ana   1    256000.0
3     juan  1    112000.0
4     ana   2    77090.0
```

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

Ejercicio 0603 - 1 punto

Dado el siguiente programa:

```
def obtieneApe(t):
  return ''.join(t.split()[-1])

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:

López
SAL
báunes
Soto
FALCÓN
pérez
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(filter(obtieneApe,nombres))`		**1**
2	`list(sorted(nombres,key=obtieneApe))`		**2**
3	`list(map(obtieneApe,nombres))`	**X**	**3**
4	`list(reversed(nombres))`		**4**

Ejercicio 0703 - 1 punto

Dado el siguiente programa:

```
print('Ingresá números naturales, 0 para terminar')
num=1
i=0
j=1
lista=[0,0,0,0,0]
sobran=[]
while num!=0:
  num=int(input(str(j)+': '))
  if num>0 and num%2==0:
    try:
      lista[i]=num
      i+=1
    except IndexError:
      sobran.append(num)
  j+=1
while 0 in lista:
  lista.remove(0)
```

Y los siguientes ingresos:

```
1: 66
2: -8
3: 4
4: 5
5: 12
6: 32
7: 40
8: 10
9: 0
```

¿Qué contenido tendrán *lista* y *sobran* al finalizar?

1	`lista: [66, 4, 12, 32, 40]` `sobran: [10]`	**X**	**1**
2	`lista: [0, 0, 0, 0, 0]` `sobran: [66, 4, 12, 32, 40, 10]`		**2**
3	`lista: [66, 4, 12, 32, 40, 10]` `sobran: []`		**3**
4	`lista: [66, 4, 12, 32, 40, 10]` `sobran: [-8, 5]`		**4**

Ejercicio 0803 - 1 punto

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

1.	`ax.set_title("Gráfico de ventas")`		**1**
2.	`ax.set_ylim(0, 7)`		**2**
3.	`ax.grid()`		**3**
4.	`ax.scatter(x, y, label='Ventas')`	**x**	**4**

Ejercicio 0903 - 1 punto

Dado el siguiente dataframe, como ejemplo se muestran las primeras 5 filas.

	local	visitante	goles local	goles visitante	posesión local	posesión visitante
0	QATAR	ECUADOR	0	2	42%	50%
1	ENGLAND	IRÁN	6	2	72%	19%
2	SENEGAL	NETHERLANDS	0	2	44%	45%
3	UNITED STATES	WALES	1	1	51%	39%
4	ARGENTINA	SAUDI ARABIA	1	2	64%	24%

Si se quiere obtener el equipo que haya convertido más de 3 goles con menos posesión en un partido.

¿ Qué métodos/funciones de pandas se deberían usar ?

1	**sort_values, head**	**X**	**1**
2	**loc, mean**		**2**
3	**value_count, loc**		**3**
4	**Ninguna de las anteriores**		**4**

Ejercicio 1003 - 2 puntos

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

```
def leer(arch):
  menor30=[]
  amigos=open(arch,'r+',encoding='utf-8')
  datos=amigos.readlines()
  amigos.close()
  for a in datos:
    a=a.strip('\n').split('-')
    edad=int(a[2])
    if edad<30:
      invitado=a[0].capitalize()+' '+a[1].capitalize()
      menor30.append(invitado+'\n')
  return menor30

def guarda(lista,arch):
  datos=open(arch,'w',encoding='utf-8')
  for amigo in lista:
    datos.write(amigo)
  datos.close()

sub30=leer('amigos.txt')
guarda(sub30,'invitados.txt')
```

Nota: El contenido de amigos.txt es el siguiente:
laura-álvarez-31
marina-ibarra-22
juan-peláez-25
ignacio-sotto-24
diego-pérez-38

Por cada línea viene nombre, apellido y edad de un amigo separado por guión

1.	**Marina Ibarra Juan Peláez Ignacio Sotto**	**X**	**1**
2.	**Laura Álvarez Marina Ibarra Juan Peláez Ignacio Sotto Diego Pérez**		**2**
3.	**22-marina Ibarra 25-juan Peláez 24-ignacio Sotto**		**3**
4.	**LAURA-DIEGO**		**4**

Ejercicio 1103 - 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.groupby('principal')['calorías'].mean()
```

1	**principal** hidratos 1500 1000 proteínas 1800 950		**1**
2	**principal** hidratos 1250.0 proteínas 1375.0	**X**	**2**
3	**Lunes, Miércoles** 2500 **Martes, Jueves** 2750		**3**
4			**4**

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