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 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

Si *monedas* es una lista que contiene lo que deben pagar los comensales de una mesa de amigos expresado en cantidad de monedas de 5; el programa debe informar en qué casos se pueden usar monedas de 10 y cuántas emplear. Considerar que el pago debe hacerse justo (no hay cambio).
¿Qué versión de programa principal sería el correcto?
Las funciones que se invocarán son las siguientes:

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

def por5(n):
  return n*5
```

Y para el contenido de ejemplo de *monedas* ([41,92,100,27,11]) debería salir:

```
Estos puedes pagar con monedas de 10
92 monedas de 5; o 46 monedas de 10
100 monedas de 5; o 50 monedas de 10
```

1
```
monedas=[41,92,100,27,11]
print('Estos puedes pagar con monedas de 10')
importes=list(map(pares,monedas))
for val in list(filter(pares,importes)):
  print(val*5,'monedas de 5; o',val//10,'monedas de 10')
```
1

2
```
monedas=[41,92,100,27,11]
print('Estos puedes pagar con monedas de 10')
importes=list(map(por5,monedas))
por10=list(filter(pares,importes))
for val in por10:
  print(val//5,'monedas de 5; o',val//10,'monedas de 10')
```
X **2**

3
```
monedas=[41,92,100,27,11]
print('Estos puedes pagar con monedas de 10')
importes=list(map(por5,monedas))
por10=list(filter(pares,importes))
for val in monedas:
  print(val//5,'monedas de 5; o',val*10,'monedas de 10')
```
3

4
```
monedas=[41,92,100,27,11]
print('Estos puedes pagar con monedas de 10')
importes=list(map(por5,monedas))
for val in list(filter(pares,monedas)):
  print(val*5,'monedas de 5; o',val//10,'monedas de 10')
```
4

Ejercicio 0202 - 1 punto

¿Qué contenido tendrá la lista *resultados* luego de ejecutar el siguiente programa?

```
def categorias(n):
  return n%10+1

def naturales(n):
  return n>0 and n==int(n)

sorteo=[-33,45,9,66.33,-7000,0,22]
validos=list(filter(naturales,sorteo))
resultados=list(map(categorias,validos))
```

1	`[-33, 6, 10, 66, -7000, 3]`		**1**
2	`[]`		**2**
3	`[6, 10, 3]`	**X**	**3**
4	`[5, 9, 2]`		**4**

Ejercicio 0302 - 1 punto

El siguiente programa permite el ingreso de números entre -10.0 y 10.0 inclusive y los carga en una lista *numeros*. La señal de salida viene dada por el número 100.

```
numeros=[]
print('Ingrese números entre -10 y 10 inclusive, 100 para salir')
i=1
num=cargaSegura(i)
while num!=100:
  numeros.append(num)
  i+=1
  num=cargaSegura(i)
```

¿Cuál de las siguientes funciones *cargaSegura()* trabajadá adecuadamente con el programa?

1

```
def cargaSegura(k):
  sale=False
  while not sale:
    try:
      cartel=str(k)+': '
      n=float(input(cartel))
      sale=not sale
    except ValueError:
      print('Espero un número entre 10 y 50, 100 para salir')
  if n!=100 and (n<-10 or n>10):
    print('Fuera de rango')
  return n
```

1

2

```
def cargaSegura(i):
  sale=False
  while not sale:
    try:
      cartel=str(k)+': '
      n=float(input(cartel))
      if n!=100 and (n<-10 or n>10):
        print('Fuera de rango')
      return n
    except ValueError:
      print('Espero un número entre 10 y 50, 100 para salir')
      sale=not sale
```

2

3

```
def cargaSegura(k):
  sale=False
  while not sale:
    try:
      cartel=str(k)+': '
      n=float(input(cartel))
      if n!=100 and (n<-10 or n>10):
        print('Fuera de rango')
      sale=False
    except ValueError:
      print('Espero un número entre 10 y 50, 100 para salir')
      return 0
```

3

4

```
def cargaSegura(k):
  sale=False
  while not sale:
    try:
      cartel=str(k)+': '
      n=float(input(cartel))
      if n!=100 and (n<-10 or n>10):
```

X

4

```
        print('Fuera de rango')
      else:
        return n
    except ValueError:
      print('Espero un número entre 10 y 50, 100 para salir')
```

Ejercicio 0402 - 1 punto

Dado el siguiente programa:

```
def carga():
  while True:
    try:
      n=int(input('Num: '))
      if n>=0 and n<=5:
        return n
    except ValueError:
      print('Espero un número entero')

def coloca(n,lista,pos):
  try:
    lista[pos]=n
    return True
  except IndexError:
    return False


numeros=[0,0,0,0,0]
i=0
print('Ingrese números entre 1 y 5 inclusive, 0 para salir')
n=carga()
while n!=0:
  ok=coloca(n,numeros,i)
  if ok:
    n=carga()
    i+=1
  else:
    n=0
```

¿Qué contendrá la lista *números* al final si se da esta serie de ingresos?

```
Ingrese números entre 1 y 5 inclusive, 0 para salir
Num: 1
Num: 1.2
Espero un número entero
Num: 6
Num: 10
Num: 1
Num: 2
Num: 0
```

1	`[0, 0, 1, 1, 2]`		**1**
2	`[1, 1.2, 6, 10, 1]`		**2**
3	`[1, 1, 2]`		**3**
4	`[1, 1, 2, 0, 0]`	**X**	**4**

Ejercicio 0502 - 1 punto

¿Qué contendrá el archivo *seleccion.txt* una vez ejecutado el siguiente programa?

```
def elige(t):
  t=t[::-1]
  return t[0].lower()!='r'

palabras=open('palabras.txt',encoding='utf-8')
pal=palabras.readlines()
palabras.close()
for i in range(len(pal)):
  pal[i]=pal[i].strip('\n')
esas=list(filter(elige,pal))
sel=open('seleccion.txt','w',encoding='utf-8')
for p in esas:
  sel.write(p+'\n')
sel.close()
```

Considerando que el contenido del archivo *palabras.txt* que se encuentra en la misma carpeta que el programa es el siguiente:

caminar
ególatra
rodar
comer
SABIDURÍA
ESTACIONAR
INCEsante
OBTENER
CONvencer
vivir
doblega
sentir
DECIR
CULPAR
actuar

1	**actuar** **CULPAR** **DECIR** **sentir** **vivir** **CONvencer** **OBTENER** **ESTACIONAR** **comer** **rodar** **caminar**		1
2	**DECIR** **sentir** **vivir**		2
3	**doblega** **incesante** **sabiduría** **ególatra**		3
4	**ególatra** **SABIDURÍA** **INCEsante** **doblega**	X	4

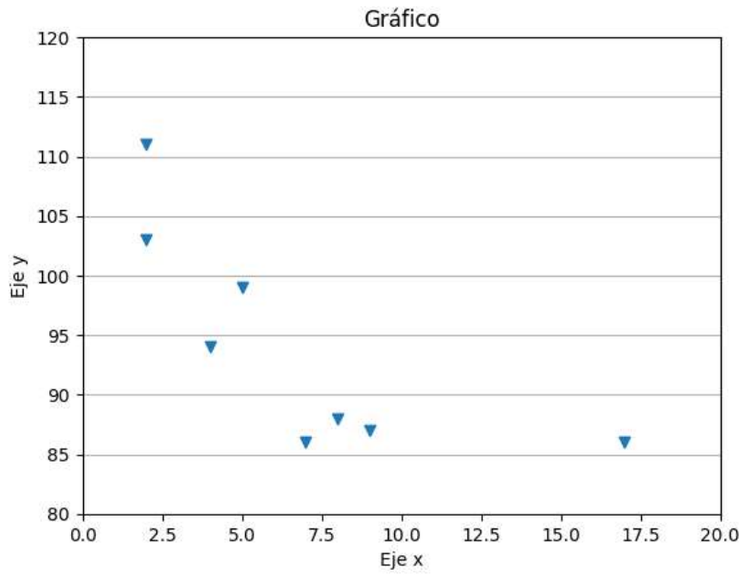
Ejercicio 0602 - 1 punto

¿Qué opción es la correcta si quiero agregar una nota al final?

```
misNotas=open('notas.txt',...,encoding='utf-8')
notas=misNotas.readlines()
nuevaNota=input('nueva: ')
misNotas.write(nuevaNota+'\n')
misNotas.close()
```

1	**'r'**		1
2	**'a'**		2
3	**'w'**		3
4	**'r+'**	X	4

Ejercicio 0702 - 1 punto



¿Cuál de las siguientes líneas de código SÍ SE CORRESPONDE con la figura?

1	**ax.set_ylim(0, 20)**		**1**
2	**ax.scatter(y, x, marker='v')**		**2**
3	**ax.grid(axis = 'y')**	**X**	**3**
4	**ax.set_ylabel('Eje x')**		**4**

Ejercicio 0802 - 1 punto

Dado el siguiente DataFrame *stock*:

	codigo	categoria	precioUnit	existencia
0	drima01	hilos	310.0	20
1	T102	hilos	250.0	35
2	Mic111	botones	425.0	180
3	ADbt012	botones	300.5	2
4	Mic154	botones	610.0	0

¿Qué se obtiene al ejecutar lo siguiente?

```
stock.groupby(stock['categoria'])['precioUnit'].mean()
```

1	categoria botones 445.166667 hilos 280.000000	**X**	**1**
2	codigo ADbt012 300.5 Mic111 425.0 Mic154 610.0 T102 250.0 drima01 310.0		**2**
3	codigo categoria precioUnit existencia 0 drima01 hilos 310.0 20 1 T102 hilos 250.0 35 2 Mic111 botones 425.0 180 3 ADbt012 botones 300.5 2		**3**
4	47.4		**4**

Ejercicio 0902 - 1 punto

```
x = [0, 1, 2, 3, 4]
x_lineal = [0, 2, 4, 6, 8]
x_bar = [0, 2, 1, 6, 3]

fig, ax = plt.subplots(nrows=3, ncols=3)
ax[1, 1].plot(x, x_lineal)
ax[1, 1].bar(x, x_bar)
plt.show()
```

¿Cuál de las siguientes líneas de código SÍ SE CORRESPONDE con la figura?

1	**Se hará una grilla de 9 gráficos. En el centro hay un gráfico de línea y en la esquina inferior derecha, uno de barras.**		**1**
2	**Se hará una grilla de 9 gráficos en donde habrá dos gráficos: uno de línea y uno de barras, ambos en la esquina superior izquierda.**		**2**
3	**Se hará una grilla de 9 gráficos en donde habrá dos gráficos: uno de línea y uno de barras, ambos en el centro.**	**X**	**3**
4	**Ninguna de las opciones describe lo que ocurre al ejecutar el código**		**4**

Ejercicio 1002 - 2 puntos

Dado el archivo lluvias.txt que contiene (en mm) la lluvia caída en una localidad durante el primer semestre de los años 2020,2021,2022,2023

```
enero;150;200;79;88
febrero;188;190;99;90
marzo;110;182;100;110
abril;66;177;112;90
mayo;60;83;80;75
junio;45;60;70;48
```

¿Qué contenido tendrá el archivo *comparativas.txt* luego de ejecutar el siguiente programa?

```
def num (val):
  return float(val)

def carga(arch):
  archivo=open(arch,encoding='utf-8')
  lineas=archivo.readlines()
  archivo.close()
  return lineas

lluvias=carga('lluvias.txt')
lluxAnio={}
for linea in lluvias:
  linea=linea.strip('\n').split(';')
  mes=linea[0]
  regAnuales=linea[1:]
  lluxAnio[mes]=regAnuales
comparativas=open('comparativas.txt','w',encoding='utf-8')
for mes in lluxAnio:
  mm=list(map(num,lluxAnio[mes]))
  comparativas.write(mes+';'+str(max(mm))+'\n')
comparativas.close()
```

1	**79.0;enero 90.0;febrero 100.0;marzo 66.0;abril 60.0;mayo 45.0;junio**		**1**
2	**enero;200.0 febrero;190.0 marzo;182.0 abril;177.0 mayo;83.0 junio;70.0**	**X**	**2**
3	**200**		**3**
4	**150;200;79;88188;190;99;90110;182;100;11066;177;112;9060;83;80;7545;60;70;48**		**4**

Ejercicio 1102 - 2 puntos

Dado el siguiente DataFrame *aberturas*:

	codigo	material	baseMetros	hMetros	vidrioMM
0	VENT01	ALUMINIO	1.30	0.7	4.0
1	VENT02	CEDRO	1.90	1.2	6.0
2	PUERTA11	ALUMINIO	0.90	2.2	NaN
3	PUERTA12	CEDRO	0.85	2.0	NaN
4	VENT03	ALUMINIO	1.00	1.0	4.0

¿Qué pasos debo seguir para obtener las medidas y los códigos de las ventanas cuya superficie sea como mínimo 1 metro?

	hMetros	baseMetros	codigo
1	1.2	1.9	VENT02
4	1.0	1.0	VENT03

1	`result=aberturas[(aberturas['codigo']>'V') &`
	`((aberturas['hMetros']*aberturas['baseMetros'])>=1)]`
	`result.loc[:,['hMetros','baseMetros','codigo']]`
		X	**1**

2	`result=aberturas[(aberturas['hMetros']>=1)]` `result.loc[:,['codigo','material','hMetros']]`		**2**
3	`result=aberturas.loc[aberturas.index[[0,3]]]` `result.loc[:,['codigo','baseMetros','hMetros']]`		**3**
4	`result=aberturas.loc[aberturas.index[[3,4]],['material','codigo']]` `result.loc[:,]`		**4**