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 0101 - 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>0 and 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 pueden 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 pueden pagar con monedas de 10')`
	`for val in list(filter(por5)).list(map(monedas,pares)):`
	`  print(val//5,'monedas de 5; o',val//10,'monedas de 10')`
			1

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

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

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

Ejercicio 0201 - 1 punto

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

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

def noNeg(n):
```

```
  return n>=0

sorteo=[-33,45,9,66,-7000,0,22]
validos=list(filter(noNeg,sorteo))
resultados=list(map(categorias,validos))
```

1	`[0, 5, 9, 6, 0, 0, 2]`		**1**
2	`[5, 9, 6, 0, 2]`	**X**	**2**
3	`[45, 9, 66, 0, 22]`		**3**
4	`[]`		**4**

Ejercicio 0301 - 1 punto

El siguiente programa permite el ingreso de números entre 10.0 y 50.0 inclusive y los carga en una lista *numeros*. La señal de salida viene dada por el número 0.

```
numeros=[]
print('Ingrese números entre 10 y 50 inclusive, 0 para salir')
i=1
num=cargaSegura(i)
while num!=0:
  numeros.append(num)
  i+=1
  num=cargaSegura(i)
```

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

1

```
def cargaSegura(i):
  sigue=True
  while sigue:
    try:
      n=float(input(str(i)+': '))
      if n==0 or (n>=10 and n<=50):
        return n
    except ValueError:
      print('Espero un número entre 10 y 50, 0 para salir')
```

X **1**

2

```
def cargaSegura(i):
  sigue=False
  while sigue:
    try:
      n=float(input(i,': '))
      if n==0 or (n>=10 and n<=50):
        return n
    except ValueError:
      print('Espero un número entre 10 y 50, 0 para salir')
```

2

3

```
def cargaSegura():
  sigue=False
  while sigue:
    try:
      cartel=str(k)+': '
      n=float(input(cartel))
      if n==0 or (n>=10 and n<=50):
        sigue=True
    except ValueError:
      print('Espero un número entre 10 y 50, 0 para salir')
  return n
```

3

4

```
def cargaSegura():
  sigue=True
  while sigue:
    try:
      cartel=str(k)+': '
      n=float(input(cartel))
      if n!=0 and (n<10 or n>50):
        print('Fuera de rango')
      sigue=False
    except ValueError:
      print('Espero un número entre 10 y 50, 0 para salir')
      sigue=False
  return n
```

4

Ejercicio 0401 - 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]
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: 1.3
Espero un número entero
Num: 1
Num: 9
Num: -1
Num: 1
Num: 6
Num: 1
Num: 5
```

1	`[1.3, 1, 9]`		**1**
2	`[1, 1, 1]`	**X**	**2**
3	`[0, 0, 0, 1, 1, 1]`		**3**
4	`[5, 1, 1, 1]`		**4**

Ejercicio 0501 - 1 punto

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

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

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	**caminar** **rodar** **comer** **ESTACIONAR** **OBTENER** **CONvencer** **vivir** **sentir** **DECIR** **CULPAR** **actuar**		**1**
2	**CAMINAR** **RODAR** **ESTACIONAR** **CULPAR** **ACTUAR**		**2**
3	**caminar** **rodar** **ESTACIONAR** **CULPAR** **actuar**	**X**	**3**
4	**COMER** **OBTENER** **CONVENCER** **VIVIR** **SENTIR** **DECIR**		**4**

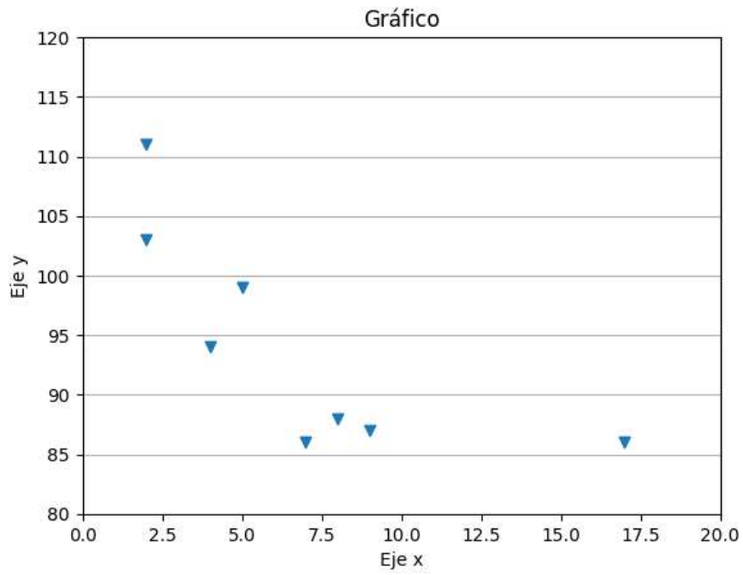
Ejercicio 0601 - 1 punto

¿Qué opción es la correcta si quiero agregar una nota al final del archivo notas.txt?

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

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

Ejercicio 0701 - 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='^')**		**2**
3	**ax.grid(axis = 'y')**	**X**	**3**
4	**ax.set_ylabel('Eje x')**		**4**

Ejercicio 0801 - 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'])['existencia'].min()
```

1	categoria botones 1335.5 182 hilos 560.0 55		**1**
2	categoria botones 0 hilos 20	**X**	**2**
3	237		**3**
4	existencia precioUnit 0 20 310.0 1 35 250.0 2 180 425.0 3 2 300.5 4 0 610.0		**4**

Ejercicio 0901 - 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[2, 2].bar(x, x_bar)
plt.show()
```

¿Cuál de las siguientes opciones describe correctamente el gráfico que se ilustrara?

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.**	**X**	**1**

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

Ejercicio 1001 - 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 *promedios.txt* luego de ejecutar el siguiente programa?

```
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
promedios=open('promedios.txt','w',encoding='utf-8')
for mes in lluxAnio:
  suma=eval('+'.join(lluxAnio[mes]))
  cant=len(lluxAnio[mes])
  promedios.write(mes+';'+str(suma/cant)+'\n')
promedios.close()
```

<u>Nota:</u> la función *eval()* devuelve el resultado de evaluar un texto que sea compatible con una expresión aritmética. Ej: eval('3+2') devuelve el número 5

1	**129.25** **141.75** **125.5** **111.25** **74.5** **55.75**		**1**
2	**150;200;79;88;129.25** **188;190;99;90;141.75** **110;182;100;110;125.5** **66;177;112;90;111.25** **60;83;80;75;74.5** **45;60;70;48;55.75**		**2**
3	**enero;129.25** **febrero;141.75** **marzo;125.5** **abril;111.25** **mayo;74.5** **junio;55.75**	**X**	**3**
4	**55.75**		**4**

Ejercicio 1101 - 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 los códigos y materiales de las aberturas que no llevan vidrio?

	codigo	material

2 PUERTA11 ALUMINIO 3 PUERTA12 CEDRO			

1	`result=aberturas[(aberturas['material']=='ALUMINIO') &`
	`(aberturas['vidrioMM']<6)]`
	`result.loc[:,]`
			1

2	`result=aberturas.iloc[2:]` `result.loc[:,['hMetros','baseMetros']]`		**2**
3	`result=aberturas.head(3)` `result.loc[:,]`		**3**
4	`result=aberturas[aberturas['vidrioMM'].isnull()]` `result.loc[:,['codigo','material']]`	**X**	**4**