PENSAMIENTO COMPUTACIONAL (90) .UBAXXI

TEMA 2

EXAMEN: SEGUNDO PARCIAL

APELLIDO:	CALIFICACIÓN:
NOMBRE:	
DNI (registrado en SIU Guaraní):	
E-MAIL:	NOTA Y FIRMA DOCENTE (no rellenar)
TEL:	
AULA:	

Duración del examen: 1:30h.

- ✔ Escribir claramente el nombre en todas las páginas.
- ✔ El examen consta de 9 preguntas de opción múltiple.
- ✔ Cada pregunta tiene una y sólo una respuesta correcta.
- ✔ Las respuestas seleccionadas deben consignarse en la siguiente matriz de opciones.
- ✔ **Sólo se considerarán las respuestas anotadas en la matriz.**
- ✔ Las preguntas de la 1 a la 5 inclusive permiten acumular 1 punto (si son correctas), de la 6 a la 9 cada una acumula 2 puntos o 0.
- ✔ La nota final se calcula de acuerdo a la siguiente función:

Puntos	1 o 2	3 o 4	5 o 6	7	8	9	10	11	12	13
Nota	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**

Matriz de Respuestas

	Ej 1 1 Pto	Ej 2 1 Pto	Ej 3 1 Pto	Ej 4 1 Pto	Ej 5 1 Pto	Ej 6 2 Ptos	Ej 7 2 Ptos	Ej 8 2 Ptos	Ej 9 2 Ptos	
1										**1**
2										**2**
3										**3**
4										**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 esta última.

Talón de Control para el Alumno

	Ej 1 1 Pto	Ej 2 1 Pto	Ej 3 1 Pto	Ej 4 1 Pto	Ej 5 1 Pto	Ej 6 2 Ptos	Ej 7 2 Ptos	Ej 8 2 Ptos	Ej 9 2 Ptos	
1										**1**
2										**2**
3										**3**
4										**4**

Ej 01 T2 – 1 Pto

Para el siguiente DataFrame alumnos:

```
      apellido  nota  ejer realizados  año carrera   prom gral
0      Giunta      10            35       1              8.75
1      Terra        2             5       2              7.00
2      Kempes      10            18       2              6.25
3      Ardiles      6            27       1              9.00
```

¿Qué instrucción produce la siguiente salida?

```
      apellido    nota  ejer realizados   año carrera prom gral
2      Kempes      10       18                  2          6.25
3      Ardiles     6        27                  1          9.00
```

1	`alumnos.sort_values('apellido')`		**1**
2	`alumnos.tail(2)`	X	**2**
3	`alumnos.groupby('año carrera')['nota'].min()`		**3**
4	`alumnos[['prom gral','nota','apellido']]`		**4**

Ej 02 T2 – 1 Pto

El archivo **Clientes.txt** contiene lo siguiente:

```
Salazar srl.;20-33456876-2
Artelac;30-36925778-4
Novallen;30-124445879-3
starluz sa.;45896985
```

¿Qué contenido tendrá el archivo **nuevo.txt** al finalizar la ejecución del siguiente programa?

```
arch=open('Clientes.txt','r')
lineas=arch.readlines()
arch.close()
arch=open('nuevo.txt','w')
for lin in lineas:
    datos=lin.split(';')
    if datos[0]!='Novallen':
        arch.write('******\n')
    else:
        arch.write(lin)
arch.close()
```

1	`******`		**1**

2	`Salazar srl.;20-33456876-2`
	`Artelac;30-36925778-4`
	`******`
	`starluz sa.;45896985`
			2

3	`Salazar srl.`
	`Artelac`
	`Novallen`
	`starluz sa.`
			3

4	`******`
	`******`
	`Novallen;30-124445879-3`
	`******`
		X	**4**

Ej 03 T2 – 1 Pto

¿Qué muestra el siguiente programa?

```
def edita(p):
    devuelve=p[0]+p[1:].lower()
    return devuelve

nom1=['ANA','CAROLINA','ESTEBAN','PEDRO']
nom2=list(map(edita,nom1))
print(*nom2)
```

Notas:
Usar un ***** antes de una lista en un print provoca que se muestran los elementos de la lista separados por blanco, sin los corchetes ni las comas
Ej:
print(*[1,2,0]) -> **1 2 0**

Usar **[desde:hasta]** con una string permite tomar porciones. Si no se pone **desde** va desde el principio y si no se pone **hasta** va hasta el final
Ejs:
'Un ejemplo común' [3:6] ->**'eje '**
'Un ejemplo común' [:6] ->**'Un eje'**

1	`a c e p`		**1**
2	`Arolina steban`		**2**
3	`aNA cAROLINA eSTEBAN pEDRO`		**3**
4	`Ana Carolina Esteban Pedro`	X	**4**

Ej 04 T2 – 1 Pto

¿Cuál es la función adecuada para el siguiente programa?

```
def traduce(…):
  -
  -
  -

traductor={'SOL':'YUAN',
           'criptomoneda':'moneda',
           'cotizará':'cerrará',
           'USDT':'u$s',
           'semana':'jornada',
           'alza':'baja',
           'al':'a la'}

txt='la criptomoneda que irá al alza esta semana será SOL. SOL cotizará a 160
USDT para el fin de semana.'
print(traduce(txt,traductor))
```

La salida del programa debe ser:

```
la moneda que irá a la baja esta jornada será YUAN. YUAN cerrará a 160 u$s
para el fin de jornada.
```

Notas:
Usar **[desde:hasta]** con una string permite tomar porciones. Si no se pone **desde** va desde el principio y si no se pone **hasta** va hasta el final
Ejs:
'Un ejemplo común' [3:6] ->**'eje '**
'Un ejemplo común' [:6] ->**'Un eje'**

1

```
def traduce(t,d):
    for cl in d:
        t=d[cl]
    return t
```

1

2

```
def traduce():
    for cl in d:
        t=t.replace(cl,d[cl])
    return cl
```

2

3

```
def traduce(t,d):
    txt=t
    for cl in d:
        if cl in txt:
            txt=txt.replace(cl,d[cl])
    return txt
```

X

3

4

```
def traduce(t):
    for cl in d:
        if cl in t:
            pos=t.index(cl)
            t=t[pos:]
    return t
```

4

Ej 05 T2 – 1 Pto

¿Cuál es el programa correcto que asegura el ingreso de un número entre 1.5 y 4.5 inclusive? El programa debe insistir en el ingreso hasta asegurarse que la variable **num** contenga un número real entre 1.5 y 4.5 inclusive.

1

```
ok=False
while not ok:
  try:
    num=float(input('Ingresa un numero entre 1.5 y
                     4.5, inclusive: '))
    if num<=4.5 and num>=1.5:
      ok=True
    else:
      print('Debe estar entre 1.5 y 4.5 inclusive')
  except ValueError:
    print('debe ser Número')
-
-
```

X **1**

2

```
while False:
    try:
        num=float(input('Ingresa un numero entre 1.5
                   y 4.5, inclusive: '))
        if num>1.5 or num<4.5:
            print('Debe estar entre 1.5 y 4.5
                     inclusive')
    except ValueError:
        print('debe ser Número')
-
-
```

2

3

```
sigue=True
while sigue:
    try:
        num=float(input('Ingresa un numero entre
                    1.5 y 4.5, inclusive: '))
        if num<1.5 and num>4.5:
            sigue=False
        else:
            print('Debe estar entre 1.5 y 4.5
                     inclusive')
    except ValueError:
        print('debe ser Número')
        sigue=False
-
-
```

3

4

```
try:
    num=float(input('Ingresa un numero entre 1.5 y
                     4.5, inclusive: '))
    if num<1.5 and num>4.5:
        print('me gusta')
    else:
        print('Debe estar entre 1.5 y 4.5 inclusive')
        num=0
except FileNotFoundError:
    print('debe ser Número')
    num=0
-
-
```

4

Ej 06 T2 – 2 Ptos

¿Qué muestra por pantalla el siguiente programa?

```
def funcion(n):
    resp=False
    if n%3!=0 and n%2==0:
        resp=True
    return resp

numeros=[3,6,7,8,15,1]
nuevaLista=list(filter(funcion,numeros))
print(nuevaLista)
```

1	`[]`		**1**
2	`[3, 15]`		**2**
3	`[1, 15, 8, 7, 6, 3]`		**3**
4	`[8]`	X	**4**

Ej 07 T2 – 2 Ptos

¿Qué debería contener la lista **`parcial`** para que el siguiente programa funcione?

```
datos={4:['juan','aguirre',301],
       1:['maría','iturre',301],
       2:['elena','lempert',13],
       3:['ignacio','lobos',11],}

parcial=[…] #elegir el contenido correcto para la salida de abajo

for valor in parcial:
    if valor in datos:
        nombre=datos[valor][0]+' '+(datos[valor][1].upper())
        aula=datos[valor][2]
    else:
        nombre=valor
        aula='No tiene aula'
    print(nombre,'Aula:',aula)
```

La salida del programa debe ser:

```
ignacio LOBOS Aula: 11
maría ITURRE Aula: 301
0 Aula: No tiene aula
```

1	`[3,1,0]`	X	**1**
2	`[]`		**2**
3	`['41444318','42235679','40996009']`		**3**
4	`[‘40356009’,'42103561','41444318','42235679']`		**4**

Ej 08 T2 – 2 Ptos

¿Qué muestra el siguiente programa?

```
def estacion(m):
  estaciones={1:'verano',2:'verano',3:'verano-otoño',
              4:'otoño',5:'otoño',6:'otoño-invierno',
              7:'invierno',8:'invierno',
              9:'invierno-primavera',
              10:'primavera',11:'primavera',
              12:'primavera-verano'}
  return 'invierno' in estaciones[m]

meses={1:'Enero',2:'Febrero',3:'Marzo',
       4:'Abril',5:'Mayo',6:'Junio',
       7:'Julio',8:'Agosto',9:'Septiembre',
       10:'Octubre',11:'Noviembre',12:'Diciembre'}

viajes=[7,11,5,4,8]
viajarEnInv=list(filter(estacion,viajes))

for m in viajarEnInv:
  print(meses[m])
```

1	7 11		**1**
2	Julio Agosto	X	**2**
3	invierno invierno		**3**
4	julio noviembre mayo abril agosto		**4**

Ej 09 T2 – 2 Ptos

Para el siguiente DataFrame alumnos:

```
      apellido  nota  ejer realizados  año carrera   prom gral
0      Giunta     10             35       1              8.75
1      Terra       2              5       2              7.00
2      Kempes     10             18       2              6.25
3      Ardiles     6             27       1              9.00
```

¿Qué salida produce la siguiente instrucción?

```
alumnos.loc[1:2,['apellido','año carrera']]
```

1	apellido nota ejer realizados año carrera prom gral 0 Giunta 10 35 1 8.75 1 Terra 2 5 2 7.00 3 Ardiles 6 27 1 9.00		**1**
2	apellido prom gral 0 Giunta 8.75 1 Terra 7.00 2 Kempes 6.25 3 Ardiles 9.00		**2**
3	apellido año carrera 1 Terra 2 2 Kempes 2	**X**	**3**
4	apellido nota ejer realizados año carrera prom gral 0 Giunta 10 35 1 8.75 3 Ardiles 6 27 1 9.00		**4**