PENSAMIENTO COMPUTACIONAL (90) .UBAXXI

EXAMEN: RECUPERATORIO SEGUNDO PARCIAL **TEMA 1**

APELLIDO:	CALIFICACIÓN:
NOMBRE:	
DNI (registrado en SIU Guaraní):	
E-MAIL:	DOCENTE (nombre y apellido):
TEL:	
AULA:	

Duración del examen: 1:30h.

- ✔ Escribir claramente el nombre en todas las páginas.
- ✔ El examen consta de 10 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 7 inclusive permiten acumular 1 punto (si son correctas), de la 8 a la 10 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 1 Pto	Ej 7 1 Pto	Ej 8 2 Ptos	Ej 9 2 Ptos	Ej 10 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.

0101 – 1 Pto

¿Con qué contenido queda la variable ***datos*** en el siguiente programa?

```
def funcion(fila):
  return fila[1]

# primos contiene nombres y edades de mis primos
primos=[['maría',12],['juan',17],['ana',16],['gonzalo',20]]
datos=list(map(funcion,primos))
mayor=max(datos)
print('El mayor de mis primos tiene',mayor,'años')
```

1	['maría','juan','ana','gonzalo']		1
2	{'maría':12,'juan':17,'ana':16,'gonzalo':20}		2
3	[12, 17, 16, 20]	X	3
4	[12]		4

0201 – 1 Pto

¿Con qué contenido queda la variable ***datos*** en el siguiente programa?

```
def funcion(fila):
  return fila[1]=='primo'

# parientes contiene nombre y vínculo de mis parientes
parientes=[['maría','tío'],['juan','primo'],
           ['ana','primo'],['gonzalo','hermano']]
datos=list(filter(funcion,parientes))
print('Primos:')
for fila in datos:
  print(fila[0])
```

La salida del programa será:

Primos:
juan
ana

1	{'primo': ['juan', 'ana']}		1
2	{'primo': ['juan', 'ana'], 'tío': 'maría', 'hermano': 'gonzalo'}		2
3	['JUAN', 'ANA']]		3
4	[['juan', 'primo'], ['ana', 'primo']]	X	4

0301 – 1 Pto

Dado el archivo ***datos.txt*** con el siguiente contenido:

lechuga 1 planta
tomates 1/2 kg
huevos 3 unid

Y el siguiente programa:

```
arch=open('datos.txt','r+')
txt=arch.readlines()
arch.close()
todasPal=[]
for linea in txt:
  linea=linea.strip('\n')
  palabras=linea.split()
  todasPal.extend(palabras)
arch=open('palabras.txt','w')
for pal in todasPal:
  arch.write(pal+'\n')
arch.close()
```

¿Qué contenido quedará en el archivo ***palabras.txt***?

Notas:
El método ***extend()*** le agrega una lista a otra, al final
Ej:
a=[1,2]
b=[0,0]
a.extend(b) -> a=[1,2,0,0]

El método ***split()*** sin argumentos arma una lista con las partes de un texto separadas por blancos
Ej:
txt='hola a todos'
lista=txt.split() -> lista=[`hola',`a',`todos']

El método ***strip()*** elimina el argumento de las puntas de un texto
Ej:
txt='—ya—`
txt.strip(`—`) -> `ya'

1	`Lechuga1planta`
	`tomates1/2kg`
	`huevos3unid`
			1

2	`lechuga`
	`tomates`
	`huevos`
			2

3	`lechuga-1-planta-tomates-1/2-kg-huevos-3-unid`		**3**

4	`lechuga`
	`1`
	`planta`
	`tomates`
	`1/2`
	`Kg`
	`huevos`
	`3`
	`Unid`
		X	**4**

0401 – 1 Pto

¿Cuál de las siguientes funciones ***abre()*** logra abrir un archivo para lectura evitando que el programa falle si el archivo de texto no existe?

```
def abre(archivo):
  -
  -
  -

#Ppal
arch=abre('xx.txt')
```

1

```
def abre(archivo):
  arch=open(archivo,'r')
  return arch
```

1

2

```
def abre(archivo):
  try:
    arch=open(archivo,'r')
  except FileNotFoundError:
    print('No está el archivo',archivo,'en la carpeta')
    print('Lo creo vacío')
    arch=open(archivo,'w')
    arch.write('')
    arch.close()
    arch=open(archivo,'r')
  return arch
```

X **2**

3

```
def abre(archivo):
  arch=open(archivo,'a')
```

3

4

```
def abre(archivo):
  if arch==open(archivo,'r'):
    modo='w'
  else:
    modo='r'
    print('No está el archivo',archivo,'en la carpeta')
    print('Lo creo vacío')
    arch=open(archivo,modo)
  return arch
```

4

0501 – 1 Pto

¿Cuál de los siguientes programas **no** llena ***categorias*** de la siguiente manera?

```
categorias={'primario': ['rojo', 'azul'],
 'valor': ['negro'],
 'secundario': ['verde', 'naranja']}
```

1
```
colores=[['rojo','primario',],['negro','valor'],
         ['verde','secundario'],['azul','primario'],
         ['naranja','secundario']]
categorias={}
for col in colores:
  categorias[col[1]]=col[0]
```
X **1**

2
```
colores=[['rojo','primario',],['negro','valor'],
         ['verde','secundario'],['azul','primario'],
         ['naranja','secundario']]
categorias={}
for col in colores:
  if col[1] in categorias:
    categorias[col[1]].append(col[0])
  else:
    categorias[col[1]]=[col[0]]
```
2

3
```
colores=[['rojo','primario',],['negro','valor'],
         ['verde','secundario'],['azul','primario'],
         ['naranja','secundario']]
categorias={}
for col in colores:
  color=col[0]
  grupo=col[1]
  if grupo in categorias:
    categorias[grupo].append('')
    ultimo=len(categorias[grupo])-1
    categorias[grupo][ultimo]=color
  else:
    categorias[grupo]=['']
    categorias[grupo][0]=color
```
3

4
```
colores=[['rojo','primario',],['negro','valor'],
         ['verde','secundario'],['azul','primario'],
         ['naranja','secundario']]
categorias={}
for col in colores:
  categorias[col[1]]=[]
for col in colores:
  categorias[col[1]].append(col[0])
```
4

0601 – 1 Pto

Dado el siguiente DataFrame ***comidas***:

```
   nombre                   categ       dificultad  origen
0  pizza                    entrada       1   italiano
1  saltimboca a la romana   principal     3          italiano
2  pastel saint honoré      postre         5   francés
3  humita                   principal     3          norteño
4  baklava                  postre         4   árabe
```

¿Qué instrucción produce la siguiente salida?

```
      nombre             categ       dificultad  origen
1     saltimboca a la romana  principal   3           italiano
```

1
```
comidas.info()
```
2
```
comidas.head(3)
```
3
```
comidas['dificultad'].sum()
```
4
```
comidas[(comidas['categ']=='principal')&
        (comidas['origen']=='italiano')]
```
X

0701 – 1 Pto

¿Cuál programa valida correctamente el ingreso de un número par? No debe dejar pasar nada que no sea un número par, impedir que el programa se interrumpa por un error de ingreso e insistir en el mismo hasta que el ingreso cumpla con lo esperado.

Nota:
La serie de los pares está formada por números naturales (enteros positivos), múltiplos de 2

1

```
num=int(input('Ingrese un número par: '))
if num> 0 and num%2==0:
  print('Ingreso Inválido')
else:
  print(num)
```

2

```
ingreso=False
while ingreso:
  try:
    num=int(input('Ingrese un número par: '))
    if num> 0 and num%2==0:
      print(num)
    else:
      print('No es un número par')
    ingreso=False
  except ValueError:
    print('Debe ser un número entero')
```

3 **X**

```
ingreso=True
while ingreso:
  try:
    num=int(input('Ingrese un número par: '))
    if num> 0 and num%2==0:
      ingreso=False
    else:
      print('No es un número par')
  except ValueError:
    print('Debe ser un número entero')
print(num)
```

4

```
ingreso=False
while not ingreso:
  try:
    num=int(input('Ingrese un número par: '))
    if num> 0 and num%2==0:
      ingreso=False
    else:
      print('No es un número par')
      ingreso=True
  except ValueError:
    print('Debe ser un número entero')
    ingreso=True
print(num)
```

0801 – 2 Ptos

Selecciona la porción de código faltante en el siguiente programa para que la salida final sea:

['córdoba', 'neuquén', 'tucumán']

Programa:

```
def minus(p):
  return p.lower()

def tilde(p):
  tildes=0
  voc='áéíóú'
  for v in voc:
    tildes+=p.count(v)
  return tildes>0

#Ppal
ciudades=['salta','CÓRDOBA','Jujuy','NEUQUÉN','tucumán']
..... #porción de código a completar
print(ciuFinal)
```

1	`ciuTilde=list(filter(tilde,ciudades))`
	`ciuMin=list(map(minus,ciuTilde))`
	`ciuFinal=ciuMin`
			1

2	`ciuTilde=list(filter(tilde,ciudades))`
	`ciuMin=list(map(minus,ciuTilde))`
	`ciuFinal=ciudades`
			2

3	`ciuMin=list(map(minus,ciudades))`
	`ciuTilde=list(filter(tilde,ciuMin))`
	`ciuFinal=ciuTilde`
		X	**3**

4	`ciuTilde=list(filter(tilde,ciudades))`
	`ciuMin=list(map(tilde,ciuTilde))`
	`ciuFinal=ciuMin`
			4

0901 – 2 Ptos

¿Cuáles debería ser los modos de apertura en cada invocación de la función open() en el siguiente programa para que no se produzca error?

```
arch=open('datos.txt', …) #primer open
txt=arch.readline()
arch.close()
print(txt)
txt1=input('Nuevo Texto: ')
arch=open('datos.txt', …) #segundo open
arch.write(txt1+'\n')
arch.close()
arch=open('datos.txt', …) #tercer open
txt=arch.readlines()
arch.close()
for linea in txt:
  print(linea.strip('\n'))
```

1	**Primer open: r** **Segundo open: a** **Tercer open: r**	**X**	**1**
2	**Primer open: w** **Segundo open: r** **Tercer open: r+**		**2**
3	**Primer open: r+** **Segundo open: r** **Tercer open: a**		**3**
4	**Primer open: r+** **Segundo open: r** **Tercer open: r+**		**4**

1001 – 2 Ptos

Dado el siguiente DataFrame ***comidas***:

```
   nombre                   categ       dificultad  origen
0  pizza                    entrada       1   italiano
1  saltimboca a la romana   principal     3          italiano
2  pastel saint honoré      postre        5   francés
3  humita                   principal     3          norteño
4  baklava                  postre        4   árabe
```

¿Qué salida produce la siguiente instrucción?

```
comidas['origen'].value_counts()
```

1	nombre categ dificultad origen 1 saltimboca a la romana principal 3 italiano 3 humita principal 3 norteño		**1**
2	origen italiano 2 francés 1 norteño 1 árabe 1	**X**	**2**
3	Dificultad categ nombre 0 1 entrada pizza 1 3 principal saltimboca a la romana 2 5 postre pastel saint honoré 3 3 principal humita 4 4 postre baklava		**3**
4	1		**4**

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 1 Pto	Ej 7 1 Pto	Ej 8 2 Ptos	Ej 9 2 Ptos	Ej 10 2 Ptos	
1											**1**
2											**2**
3											**3**
4											**4**