| | |
|---|---|
| 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 0107 - 1 punto**

**¿Qué contiene *b* ?**

```
def organiza(n):
  if n%3==0 and n<100 and n%2==0:
    return True
  else:
    return False

a=[15,-150,150,66]
b=list(filter(organiza,a))
```

| | | | |
|---|---|---|---|
| **1.** | **[66, -150]** | | **1** |
| **2.** | **[-150, 66]** | **X** | **2** |
| **3.** | **[150, 66]** | | **3** |
| **4.** | **[]** | | **4** |

**Ejercicio 0207 - 1 punto**

**¿Cuál versión de la función ingreso() valida correctamente los datos de acceso a una cuenta de mail? Deben coincidir dirección de mail y password**

```
def ingreso(usr):

def ingreso(usr):
   -
   -

# users contiene dirección de mail:password
users={'info@usina.com.ar':'X25fc230',
       'iro@hotmail.com':'Irma1211',
       'ergo@gmail.com':'joacoS01'}
while not ingreso(users):
  print('Datos de Acceso Erróneos')
```

| | | | |
|---|---|---|---|
| **1.** | `def ingreso(usr):`<br>`  dccMail=input('Mail: ')`<br>`  clave=input('Password: ')`<br>`  if dccMail in usr and clave==usr[dccMail]:`<br>`  return True` | **X** | **1** |
| **2.** | `def ingreso(usr):`<br>`  dccMail=input('Mail: ')`<br>`  clave=input('Password: ')`<br>`  if clave==usr[dccMail].lower():`<br>`  return True` | | **2** |

| | | | |
|---|---|---|---|
| **3.** | `def ingreso(usr):`<br>`  dccMail=input('Mail: ')`<br>`  clave=input('Password: ')`<br>`  i=0`<br>`  resp=False`<br>`  for mail in usr:`<br>`  if dccMail==mail:`<br>`  if usr[i]==clave:`<br>`  resp=True`<br>`  i+=1`<br>`  return resp` | | **3** |
| **4.** | `def ingreso(usr):`<br>`  dccMail=input('Mail: ').lower()`<br>`  clave=input('Password: ')`<br>`  resp=True`<br>`  if dccMail in usr:`<br>`  if usr[dccMail]==clave:`<br>`  resp=False`<br>`  return resp` | | **4** |

**Ejercicio 0307 - 1 punto**

**¿Cuál versión de la función abreParaLeer() evita la interrupción del programa por un fallo de archivo no encontrado en el caso de que el archivo no se encuentre?**
**Nota: Se espera que la función evite la parada del programa por ese fallo y eventualmente cree un archivo nuevo, sólo si no existe**

```
def abreParaLeer(arch):
  -
  -

datos=abreParaLeer('uno.txt')
-
-
datos.close()
```

| | | | |
|---|---|---|---|
| **1.** | `def abreParaLeer(arch):`<br>`  try:`<br>`  a=open(arch,'r')`<br>`  return a`<br>`  except FileNotFoundError:`<br>`  print('No existe el archivo')` | | **1** |
| **2.** | `def abreParaLeer(arch):`<br>`  a=open(arch,'w')`<br>`  a.close()`<br>`  try:`<br>`  a=open(arch,'r')`<br>`  except FileNotFoundError:`<br>`  a=open(arch,'w')`<br>`  return a` | | **2** |
| **3.** | `def abreParaLeer(arch):`<br>`  try:`<br>`  a=open(arch)`<br>`  except FileNotFoundError:`<br>`  a=open(arch,'w')`<br>`  a.close()`<br>`  a=open(arch)`<br>`  return a` | **X** | **3** |
| **4.** | `def abreParaLeer(arch):`<br>`  a=open(arch,'r+')`<br>`  try:`<br>`  a=open(arch,'a')`<br>`  except FileNotFoundError:`<br>`  a=open(arch,'r')`<br>`  return a` | | **4** |

**Ejercicio 0407 - 1 punto**

**¿Cuál versión de la función guarda() genera un archivo personal.txt que contendrá únicamente los datos que le pasa el programa? El archivo debe quedar con el siguiente formato y contenido:**
**ana paz,AP**
**inés alza,IA**
**sergio ortiz,SO**

```
def guarda(dicci,arch):
  -
  -

nombres={'ana':'paz','inés':'alza','sergio':'ortiz'}
guarda(nombres,'personal.txt')
```

| | | | |
|---|---|---|---|
| **1.** | `def guarda(dicci,arch):`<br>`  dat=open(arch,'r',encoding='utf-8')`<br>`  lineas=[]`<br>`  for nom in dicci:`<br>`  ape=dicci[nom]`<br>`  lin=ape[0]+nom[0]+','+ape.upper()+' '+nom.upper()`<br>`  lineas.append(lin)`<br>`  dat.writelines(lineas)`<br>`  dat.close()` | | **2** |
| **2.** | `def guarda(dicci,arch):`<br>`  dat=open(arch,'w',encoding='utf-8')`<br>`  for i in range(len(dicci)):`<br>`  ape=dicci[i]`<br>`  dat.write(ape[0]+ape[1]+','+ape[0].upper()+ape[1].upper())`<br>`  arch.close()` | | **3** |
| **3.** | `def guarda(d,arch):`<br>`  datos=open(arch,'a',encoding='utf-8')`<br>`  for nom in d:`<br>`  ape=d[nom]`<br>`  datos.write(nom[0]+ape[1]+','+nom+ape)`<br>`  datos.close()` | | **4** |
| **4.** | `def guarda(dicci,arch):`<br>`  dat=open(arch,'w',encoding='utf-8')`<br>`  for nom in dicci:`<br>`  ape=dicci[nom]`<br>`  dat.write(nom+' '+ape+','+nom[0].upper()+ape[0].upper()+'\n')`<br>`  dat.close()` | **X** | |

**Ejercicio 0507 - 1 punto**

**Dado el siguiente DataFrame *lluvias*:**

| | localidad | mes | mm |
|---|---|---|---|
| 0 | azul | abril | 65 |
| 1 | charata | abril | 96 |
| 2 | azul | julio | 30 |
| 3 | azul | octubre | 72 |
| 4 | federación | enero | 110 |
| 5 | quitilipi | marzo | 120 |
| 6 | federación | marzo | 115 |

**Que contiene 7 filas y 3 columnas: localidad, mes y precipitación total registrada en mm (mm).**

**¿Qué instrucción produce la siguiente salida?**

```
      localidad  mes      mm
0      azul       abril    65
2      azul       julio    30
3      azul       octubre   72
```

| | | | |
|---|---|---|---|
| **1.** | `lluvias[(lluvias['localidad']=='azul')]` | **X** | **1** |
| **2.** | `lluvias.head()` | | **2** |
| **3.** | `lluvias.sum()` | | **3** |
| **4.** | `lluvias.loc[lluvias.index[[0,2,3]],['mm']].sum()` | | **4** |

**Ejercicio 0607 - 1 punto**

**Dado el siguiente programa:**

```
def obtieneNom(t):
  return 'ez' in t.lower().split()[-1]

nombres=['ana López','emiliano SAL','LORENA ana báunes',
         'analía Soto','ángelea FALCÓN',
         'Luciana analía pérez','Ana María Giménez']
nombres.sort()
resultado= . . .
for nom in resultado:
  print(nom)
```

**Que produce la siguiente salida:**

```
Ana María Giménez
Luciana analía pérez
ana López
>>>
```

**¿Qué instrucción debería ir en los puntos suspensivos?**

**Nota: 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(map(obtieneNom,nombres))` | | **1** |
| **2.** | `reversed(nombres)` | | **2** |
| **3.** | `obtieneNom(nombres)` | | **3** |
| **4.** | `list(filter(obtieneNom,nombres))` | **X** | **4** |

**Ejercicio 0707 - 1 punto**

**Dado el siguiente programa:**

```
print('Elimina Elementos de')
nombres=['julio','ana','inés','juan','lía']
print(nombres)
sigue=True
respSi=('s','si','sipi','sisi','oki','dale')
while sigue:
  opc=input('Saca? (si/no) ')
  if opc.lower() in respSi:
    try:
      elimina=nombres.pop()
    except IndexError:
      sigue=False
  else:
    sigue=False
```

**Y los siguientes ingresos:**

```
Elimina Elementos de
['julio', 'ana', 'inés', 'juan', 'lía']
Saca? (si/no) dale
Saca? (si/no) SIpi
Saca? (si/no) nono
```

**¿Qué contenido tendrá *nombres* al finalizar?**

| | | | |
|---|---|---|---|
| **1.** | `[]` | | **1** |
| **2.** | `['julio', 'ana', 'inés', 'juan', 'lía']` | | **2** |
| **3.** | `['ana', 'inés', 'juan', 'lía']` | | **3** |
| **4.** | `['julio', 'ana', 'inés']` | **X** | **4** |

**Ejercicio 0807 - 1 punto**


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

| | | | |
|---|---|---|---|
| **1.** | `ax.set_xlim(0, 8)` | X | **1** |
| **2.** | `ax.grid()` | | **2** |
| **3.** | `ax.set_ylabel('Ventas ($)')` | | **3** |
| **4.** | `ax.set_title("Gráfico de ventas")` | | **4** |

**Ejercicio 0907 - 1 punto**

**Dado el siguiente programa, ¿ cómo queda el archivo luego de cada ejecución ?**

```
archivo=open('agenda.txt')
contactos=archivo.readlines()
archivo.close()

apellido=input('Ingresá Apellido del nuevo Contacto: ')
nombre=input(f'Nombre de {apellido}: ')
celular=input(f'Celular de {nombre} {apellido}: ')
nuevo=apellido +' '+ nombre +', '+ celular +'\n'
contactos.extend(nuevo)
archivo=open('agenda.txt', 'w')
archivo.writelines(contactos)
archivo.close()
```

| | | | |
|---|---|---|---|
| **1.** | **Con una línea** | | **1** |
| **2.** | **Se agrega una nueva línea** | **X** | **2** |
| **3.** | **Se reemplaza la primer línea** | | **3** |
| **4.** | **Ninguna de las anteriores** | | **4** |

**Ejercicio 1007 - 2 puntos**

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

```
def leer(arch):
  receta=open(arch,encoding='utf-8')
  instrucciones=receta.readlines()
  receta.close()
  ingrePesar=list(filter(soloPeso,instrucciones))
  return ingrePesar

def soloPeso(txt):
  return 'gr\n' in txt

def guarda(lista,arch):
  datos=open(arch,'w',encoding='utf-8')
  for ingre in lista:
    desc=ingre.strip('gr\n').upper()
    datos.write(desc+'gr\n')
  datos.close()

aPesar=leer('receta.txt')
guarda(aPesar,'sePesa.txt')
```

**Nota: El contenido de receta.txt es el siguiente:**

**Ingredientes**
**harina 200 gr**
**fécula 300 gr**
**yemas 3 unid**
**manteca 200 gr**
**azúcar 150 gr**
**dulce de leche c/n**
**esencia de vainilla 1 chdita**
**ralladura de limón 1 chdita**
**coco rallado c/n**
**polvo de hornear 1 chdita**

**Preparación**
**Cremar manteca con azúcar - Integrar las yemas - Saborizar - Incorporar secos - Estirar masa - Cortar discos - Hornear**

| | | | |
|---|---|---|---|
| **1.** | **harina 200 gr<br>fécula 300 gr<br>yemas 3 unid<br>manteca 200 gr<br>azúcar 150 gr<br>dulce de leche c/n<br>esencia de vainilla 1 chdita<br>ralladura de limón 1 chdita<br>coco rallado c/n<br>polvo de hornear 1 chdita** | | **1** |
| **2.** | **HARINA 200 gr<br>FÉCULA 300 gr<br>MANTECA 200 gr<br>AZÚCAR 150 gr** | **X** | **2** |
| **3.** | **harina 200 gr fécula 300 gr manteca 200 gr azúcar 150 gr** | | **3** |
| **4.** | **YEMAS 3<br>DULCE DE LECHE c/n<br>ESENCIA DE VAINILLA 1<br>RALLADURA DE LIMÓN 1<br>COCO RALLADO c/n<br>POLVO DE HORNEAR 1** | | **4** |

**Ejercicio 1107 - 2 puntos**

**Dado el siguiente DataFrame *cobertura*:**

| | región | provincia | sucursales |
|---|---|---|---|
| 0 | cuyo | san juan | 3 |
| 1 | litoral | santa fé | 0 |
| 2 | cuyo | mendoza | 5 |
| 3 | patagonia | chubut | 2 |
| 4 | cuyo | san luis | 1 |
| 5 | pampa | buenos aires | 8 |

**Que contiene 6 filas y 3 columnas: región geográfica (región), provincia y cantidad de sucursales abiertas (sucursales).**

**¿Qué se muestra como resultado al ejecutar las siguientes instrucciones?**

```
cobertura=cobertura.drop(1)
cobertura['región'].value_counts()
```

| | | | |
|---|---|---|---|
| **1.** | cuyo 3<br>patagonia 1<br>pampa 1<br>santa fé 1 | | **1** |
| **2.** | cuyo 3<br>patagonia 1<br>pampa 1 | **X** | **2** |
| **3.** | cuyo 9<br>patagonia 2<br>pampa 8 | | **3** |
| **4.** | cuyo 3 | | **4** |