Técnicas de Programación

Instituto de Formación Técnica Superior Nro. 11
Docente: Lic. Norberto A. Orlando

Laboratorio UNIDAD 4: Funciones, Tuplas, Diccionarios, Excepciones y Procesamiento de Datos

Ejercicio 1

¿Cuál es la salida de los siguientes fragmentos de código?

a)

```
def intro(a="James Bond", b="Bond"):
     print("Mi nombre es", b + ".", a + ".")

intro()
```

b)

```
def intro(a="James Bond", b="Bond"):
    print("Mi nombre es", b + ".", a + ".")

intro(b="Sean Connery")

```

c)

```
def intro(a, b="Bond"):
    print("Mi nombre es", b + ".", a + ".")

intro("Susan")

```

d)

```python
def add_numbers(a, b=2, c):
    print(a + b + c)

add_numbers(a=1, c=3)

```

Ejercicio 2

¿Cuál es la salida de los siguientes fragmentos de código?

a)

```python
def hi():
    return
    print("¡Hola!")

hi()
```

b)

```python
def is_int(data):
    if type(data) == int:
        return True
    elif type(data) == float:
        return False

print(is_int(5))
print(is_int(5.0))
print(is_int("5"))

```

c)

```python
def even_num_lst(ran):
    lst = []
    for num in range(ran):
        if num % 2 == 0:
            lst.append(num)
    return lst

print(even_num_lst(11))

```

d)

```python
def list_updater(lst):
    upd_list = []
    for elem in lst:
        elem **= 2
        upd_list.append(elem)
    return upd_list

foo = [1, 2, 3, 4, 5]
print(list_updater(foo))

```

Ejercicio 3

Se solicita escribir y probar una función que toma un argumento (un año) y devuelve True si el año es un año bisiesto, o False si no lo es.

Parte del esqueleto de la función ya está en el editor.

Nota: también hemos preparado un breve código de prueba, que puedes utilizar para probar tu función.

El código utiliza dos listas - una con los datos de prueba y la otra con los resultados esperados. El código te dirá si alguno de tus resultados no es válido.

```python
# def is_year_leap(year):
    #
    # Escribe tu código aquí.
    #

test_data = [1900, 2000, 2016, 1987]
test_results = [False, True, True, False]
for i in range(len(test_data)):
    yr = test_data[i]
    print(yr,"->",end="")
    result = is_year_leap(yr)
    if result == test_results[i]:
        print("OK")
    else:
        print("Fallido")
```

Ejercicio 4

Se solicita escribir y probar una función que toma dos argumentos (un año y un mes) y devuelve el número de días del mes/año dado (mientras que solo febrero es sensible al valor year, tu función debería ser universal).

La parte inicial de la función está lista. Ahora, haz que la función devuelva None si los argumentos no tienen sentido.

Hemos preparado un código de prueba. Amplíalo para incluir más casos de prueba.

```python
def is_year_leap(year):
    #
    # Tu código del LAB anterior.
    #
```

```
def days_in_month(year, month):
    #
    # Escribe tu código nuevo aquí.
    #

test_years = [1900, 2000, 2016, 1987]
test_months = [2, 2, 1, 11]
test_results = [28, 29, 31, 30]
for i in range(len(test_years)):
    yr = test_years[i]
    mo = test_months[i]
    print(yr, mo, "->", end="")
    result = days_in_month(yr, mo)
    if result == test_results[i]:
        print("OK")
    else:
        print("Fallido")
```

Ejercicio 5

Se solicita escribir y probar una función que toma tres argumentos (un año, un mes y un día del mes) y devuelve el día correspondiente del año, o devuelve None si cualquiera de los argumentos no es válido.

Debes utilizar las funciones previamente escritas y probadas del ejercicio 3 y 4.

Agrega algunos casos de prueba al código.

```
def is_year_leap(year):
    #
    # Tu código del ejercicio 3
    #

def days_in_month(year, month):
    #
    # Tu código del ejercicio 4
    #

def day_of_year(year, month, day):
    #
    # Escribe tu código nuevo aquí.
    #

print(day_of_year(2000, 12, 31))
```

Ejercicio 6

Un número natural es primo si es mayor que 1 y no tiene divisores más que 1 y si mismo.

Por ejemplo, 8 no es un número primo, ya que puedes dividirlo entre 2 y 4 (no podemos usar divisores iguales a 1 y 8, ya que la definición lo prohíbe).

Por otra parte, 7 es un número primo, ya que no podemos encontrar ningún divisor para el.

Tu tarea es escribir una función que verifique si un número es primo o no.

La función:

- se llama is_prime;
- toma un argumento (el valor a verificar)
- devuelve True si el argumento es un número primo, y False de lo contrario.

*Sugerencia: intenta dividir el argumento por todos los valores posteriores (comenzando desde 2) y verifica el resto - si es cero, tu número no puede ser un número primo; analiza cuidadosamente cuándo deberías detener el proceso utilizando la raíz cuadrada de cualquier valor pasado, puedes utilizar el operador **.*
*Recuerda: la raíz cuadrada de x es lo mismo que x 0.5 (num ** 0.5)*

Complementa el código en el editor.
Ejecuta tu código y verifica si tu salida es la misma que la nuestra.

```
def is_prime(num):
    #
    # Escribe tu código aquí.
    #

for i in range(1, 20):
    if is_prime(i + 1):
        print(i + 1, end=" ")
print()
```

Ejercicio 7

El consumo de combustible de un automóvil se puede expresar de muchas maneras diferentes. Por ejemplo, en Europa, se muestra como la cantidad de combustible consumido por cada 100 kilómetros. En los EE. UU., se muestra como la cantidad de millas recorridas por un automóvil con un galón de combustible.
Tu tarea es escribir un par de funciones que conviertan l/100km a mpg (milas por galón), y viceversa.

Las funciones:
se llaman liters_100km_to_miles_gallon y miles_gallon_to_liters_100km respectivamente;
toman un argumento (el valor correspondiente a sus nombres)
Complementa el código en el editor y ejecuta tu código y verifica si tu salida es la misma que la nuestra.

Información para ayudarte:
1 milla = 1609.344 metros.
1 galón = 3.785411784 litros.
Salida esperada:

```
60.31143162393162
31.36194444444444
23.52145833333333
3.9007393587617467
7.490910297239916
10.009131205673757
```

```
def liters_100km_to_miles_gallon(liters):
    #
    # Escribe tu código aquí.
    #

def miles_gallon_to_liters_100km(miles):
    #
    # Escribe tu código aquí.
    #

print(liters_100km_to_miles_gallon(3.9))
print(liters_100km_to_miles_gallon(7.5))
print(liters_100km_to_miles_gallon(10.))
print(miles_gallon_to_liters_100km(60.3))
print(miles_gallon_to_liters_100km(31.4))
print(miles_gallon_to_liters_100km(23.5))
```

Ejercicio 8

¿Cuál es la salida de los siguientes fragmentos de código?

a)

```
def message():
    alt = 1
    print("¡Hola, mundo!")


print(alt)

```

b)

```
a = 1


def fun():
    a = 2
    print(a)


fun()
print(a)

```

c)

```
a = 1


def fun():
    global a
    a = 2
    print(a)


fun()
a = 3
print(a)

```

d)

```
a = 1


def fun():
    global a
    a = 2
    print(a)


a = 3
fun()
print(a)

```

Ejercicio 9

¿Cuál es la salida del siguiente fragmento de código?

```
def fun(a):
    if a > 30:
        return 3
    else:
        return a + fun(a + 3)


print(fun(25))

```

Ejercicio 10

¿Qué ocurrirá cuando se intente ejecutar el siguiente código?

```
my_tup = (1, 2, 3)
print(my_tup[2])

```

Ejercicio 11

¿Cuál es la salida del siguiente fragmento de código?

```
tup = 1, 2, 3
a, b, c = tup

print(a * b * c)

```

Ejercicio 12

Completa el código para emplear correctamente el método count() para encontrar la cantidad de 2 duplicados en la tupla siguiente.

```
tup = 1, 2, 3, 2, 4, 5, 6, 2, 7, 2, 8, 9
duplicates = # Escribe tu código aquí.

print(duplicates) # salida: 4

```

Ejercicio 13

Escribe un programa que "una" los dos diccionarios (d1 y d2) para crear uno nuevo (d3).

```
d1 = {'Adam Smith': 'A', 'Judy Paxton': 'B+'}
d2 = {'Mary Louis': 'A', 'Patrick White': 'C'}
d3 = {}

for item in (d1, d2):
    # Escribe tu código aquí.

print(d3)
```

Ejercicio 14

Escribe un programa que convierta la lista my_list en una tupla.

```python
my_list = ["car", "Ford", "flower", "Tulip"]

t = # Escribe tu código aquí.
print(t)
```

Ejercicio 15

Escribe un programa que convierta la tupla colors en un diccionario.

```python
colors = (("green", "#008000"), ("blue", "#0000FF"))

# Escribe tu código aquí.

print(colors_dictionary)

```

Ejercicio 16

¿Que ocurrirá cuando se ejecute el siguiente código?

```python
my_dictionary = {"A": 1, "B": 2}
copy_my_dictionary = my_dictionary.copy()
my_dictionary.clear()

print(copy_my_dictionary)
```

Ejercicio 17

¿Cuál es la salida del siguiente fragmento de código?

```python
colors = {
    "blanco": (255, 255, 255),
    "gris": (128, 128, 128),
    "rojo": (255, 0, 0),
    "verde": (0, 128, 0)
    }

for col, rgb in colors.items():
    print(col, ":", rgb)

```

Ejercicio 18

Desarrollar un programa con dos funciones. La primer solicite el ingreso de un entero y muestre el cuadrado de dicho valor. La segunda que solicite la carga de dos valores y muestre el producto de los mismos. LLamar desde el bloque del programa principal a ambas funciones.

Ejercicio 19
Confeccionar una función que reciba tres enteros y nos muestre el mayor de ellos. La carga de los valores hacerlo por teclado.

Ejercicio 20
Desarrollar un programa que permita ingresar el lado de un cuadrado. Luego preguntar si quiere calcular y mostrar su perímetro o su superficie.

Ejercicio 21
Desarrollar una funcion que reciba un string como parametro y nos muestre la cantidad de vocales. Llamarla desde el bloque principal del programa

Ejercicio 22
Elaborar una función que nos retorne el perímetro de un cuadrado pasando como parámetros el valor de un lado

Ejercicio 23
Definir por asignación una lista de enteros en el bloque principal del programa. Elaborar tres funciones, la primera recibe la lista y retorna la suma de todos sus elementos, la segunda recibe la lista y retorna el mayor valor y la última recibe la lista y retorna el menor.

Ejercicio 24
Crear una lista de enteros por asignación. Definir una función que reciba una lista de enteros y un segundo parámetro de tipo entero. Dentro de la función mostrar cada elemento de la lista multiplicado por el valor entero enviado.
Ejemplo:
lista=[3, 7, 8, 10, 2]
multiplicar(lista,3)

Ejercicio 25
Confeccionar una función que cargue por teclado una lista de 5 enteros y la retorne. Una segunda función debe recibir una lista y mostrar todos los valores mayores a 10. Desde el bloque principal del programa llamar a ambas funciones.

Ejercicio 26
Con funciones, desarrollar un programa que permita cargar 5 nombres de personas y sus edades respectivas. Luego de realizar la carga por teclado de todos los datos imprimir los nombres de las personas mayores de edad (mayores o iguales a 18 años)
Imprimir la edad promedio de las personas.

Ejercicio 27
En una empresa se almacenaron los sueldos de 10 personas.
Desarrollar las siguientes funciones y llamarlas desde el bloque principal:
1) Carga de los sueldos en una lista.
2) Impresión de todos los sueldos.

3) Cuántos tienen un sueldo superior a $4000.
4) Retornar el promedio de los sueldos.
5) Mostrar todos los sueldos que están por debajo del promedio.

Ejercicio 28
Desarrollar una aplicación que permita ingresar por teclado los nombres de 5 artículos y sus precios.
Definir las siguientes funciones:
1) Cargar los nombres de articulos y sus precios.
2) Imprimir los nombres y precios.
3) Imprimir el nombre de artículo con un precio mayor
4) Ingresar por teclado un importe y luego mostrar todos los artículos con un precio menor igual al valor ingresado.

Ejercicio 29
Confeccionar un programa que permita:
1) Cargar una lista de 10 elementos enteros.
2) Generar dos listas a partir de la primera. En una guardar los valores positivos y en otra los negativos.
3) Imprimir las dos listas generadas.

Ejercicio 30
Elaborar una función que muestre la tabla de multiplicar del valor que le enviemos como parámetro.
Definir un segundo parámetro llamado termino que por defecto almacene el valor 10. Se deben mostrar tantos términos de la tabla de multiplicar como lo indica el segundo parámetro.
Llamar a la función desde el bloque principal de nuestro programa con los siguientes argumentos nombrados: tabla de 3 con 5 terminos y la tabla del 3 con 20 terminos.

Ejercicio 31
Confeccionar un programa con las siguientes funciones:
1)Cargar una lista de 5 enteros.
2)Retornar el mayor y menor valor de la lista mediante una tupla.
Desempaquetar la tupla en el bloque principal y mostrar el mayor y menor.

Ejercicio 32
Almacenar en una lista 5 empleados, cada elemento de la lista es una sub lista con el nombre del empleado junto a sus últimos tres sueldos (estos tres valores en una tupla)
El programa debe tener las siguientes funciones:
1)Carga de los nombres de empleados y sus últimos tres sueldos.
2)Imprimir el monto total cobrado por cada empleado.
3)Imprimir los nombres de empleados que tuvieron un ingreso trimestral mayor a 10000 en los últimos tres meses.
Tener en cuenta que la estructura de datos si se carga por asignación debería ser similar a:

```
empleados = [["juan",(2000,3000,4233)] , ["ana",(3444,1000,5333)] ,  etc.   ]
```

Ejercicio 33

En el bloque principal del programa definir un diccionario que almacene los nombres de paises como clave y como valor la cantidad de habitantes. Implementar una función para mostrar cada clave y valor.

Ejercicio 34

Crear un diccionario que permita almacenar 5 artículos, utilizar como clave el nombre de productos y como valor el precio del mismo.
Desarrollar además las funciones de:
1) Imprimir en forma completa el diccionario
2) Imprimir solo los artículos con precio superior a 100.

Ejercicio 35

Crear un diccionario en Python que defina como clave el número de documento de una persona y como valor un string con su nombre.
Desarrollar las siguientes funciones:
1) Cargar por teclado los datos de 4 personas.
2) Listado completo del diccionario.
3) Consulta del nombre de una persona ingresando su número de documento.

Ejercicio 36

¿Cuál de las siguientes líneas inicia correctamente una definición de función sin parámetros?

a) def fun():
b) def fun:
c) function fun():
d) fun function():

Ejercicio 37

Una función definida de la siguiente manera: (Elegir dos respuestas)

a) puede ser invocada sin ningún argumento.
b) puede ser invocado con exactamente un argumento.
c) debe ser invocada con exactamente un argumento.
d) debe invocarse sin ningún argumento.

Ejercicio 38

Una función integrada es una función que:

a) viene con Python, y es una parte integral de Python
b) ha sido colocado dentro de tu código por otro programador
c) tiene que ser importado antes de su uso
d) está oculto a los programadores

Ejercicio 39

El hecho de que las tuplas pertenezcan a tipos de secuencia significa que:

a) se pueden indexar y rebanar como las listas
b) se pueden extender usando el método .append()
c) se pueden modificar usando la instrucción del
d) en realidad son listas

Ejercicio 40

¿Cuál es la salida del siguiente fragmento de código?

```
def f(x):
    if x == 0:
        return 0
    return x + f(x - 1)


print(f(3))
```

a) 6
b) 3
c) 1
d) el codigo es erroneo

Ejercicio 41

¿Cuál es la salida del siguiente fragmento de código?

```
def fun(x):
    x += 1
    return x


x = 2
x = fun(x + 1)
print(x)
```

a) 4
b) 5
c) 3
d) el código es erroneo

Ejercicio 42

¿Qué código insertarías en lugar del comentario para obtener el resultado esperado?

Salida esperada:

a
b
c

```
dictionary = {}
my_list = ['a', 'b', 'c', 'd']

for i in range(len(my_list) - 1):
    dictionary[my_list[i]] = (my_list[i], )

for i in sorted(dictionary.keys()):
    k = dictionary[i]
    # Inserta tu código aquí.
```

a) print(k[0])
b) print(k['0'])
c) print(k)
d) print(k["0"])

Ejercicio 43

El siguiente fragmento de código:

```
def func(a, b):
    return a ** a


print(func(2))
```

a) es erroneo
b) dará como salida 4
c) dará como salida 2
d) devolverá None

Ejercicio 44

¿Cuál es la salida del siguiente fragmento de código?

```
def func_1(a):
    return a ** a


def func_2(a):
    return func_1(a) * func_1(a)


print(func_2(2))
```

a) dará como salida 16
b) dará como salida 4
c) dará como salida 2
d) es erroneo

Ejercicio 45

¿Cuál de las siguientes líneas inicia correctamente una función utilizando dos parámetros, ambos con valores predeterminados de cero?

a) def fun(a=0, b=0):
b) def fun(a=b=0):
c) fun fun(a=0, b):
d) fun fun(a, b=0):

Ejercicio 46

¿Cuál es la salida del siguiente fragmento de código?

```
def fun(x):
    global y
    y = x * x
    return y


fun(2)
print(y)
```

a) 4
b) 2
c) Ninguno
d) el código provocará un error de tiempo de ejecución

Ejercicio 47

¿Cuál es la salida del siguiente fragmento de código?

```
def fun(x, y, z):
    return x + 2 * y + 3 * z


print(fun(0, z=1, y=3))
```

a) 9
b) 0
c) 3
d) el código es erroneo

Ejercicio 48

¿Cuál es la salida del siguiente fragmento de código?

```
def fun(inp=2, out=3):
    return inp * out


print(fun(out=2))
```

a) 4
b) 6
c) 2
d) el código es erroneo

Ejercicio 49

¿Cuál es la salida del siguiente fragmento de código?

```
tup = (1, 2, 4, 8)
tup = tup[1:-1]
tup = tup[0]
print(tup)
```

a) 2
b) (2)
c) (2,)
d) el código es erroneo

Ejercicio 50

Selecciona las sentencia **true** sobre el bloque try-except en relación con el siguiente ejemplo. (Selecciona dos respuestas).

```
try:
    # Algo de código...
except:
    # Algo de código...
```

a) Si sospechas que un fragmento de código puede generar una excepción, se debe colocar dentro del bloque try.
b) El código que sigue a la sentencia except será ejecutado si el código en el bloque try se encuentra con un error.
c) Si existe un error de sintaxis en el código ubicado en el bloque try, la sentencia except no lo manejará, y una excepción SyntaxError será generada.
d) El código que sigue a la sentencia try será ejecutado si el código dentro de la sentencia except se encuentra con un error.

Ejercicio 51

¿Cuál es la salida del siguiente fragmento de código?

```
try:
    value = input("Ingresa un valor: ")
    print(value/value)
except ValueError:
    print("Entrada incorrecta...")
except ZeroDivisionError:
    print("Entrada errónea...")
except TypeError:
    print("Entrada muy errónea...")
except:
    print("¡Buuu!")
```

a) Entrada muy errónea...
b) Entrada errónea...
c) Entrada incorrecta...
d) ¡Buuu!

Ejercicio 52

¿Cuál es la salida del siguiente fragmento de código?

```
my_list = [1, 2]

for v in range(2):
    my_list.insert(-1, my_list[v])

print(my_list)
```

a) [1, 2, 2, 2]
b) [2, 1, 1, 2]
c) [1, 2, 1, 2]
d) [1, 1, 1, 2]

Ejercicio 53

¿Cuál es la salida del siguiente fragmento de código?

```
def function_1(a):
    return None


def function_2(a):
    return function_1(a) * function_1(a)


print(function_2(2))
```

a) dará como salida 16
b) provocará un error de ejecución
c) dará como salida 4
d) dará como salida 2

Ejercicio 54

¿El resultado de la siguiente division?

```
1 | 1 // 2
```

a) es igual a 0.0
b) no se puede predecir
c) es igual a 0
d) es igual a 0.5

Ejercicio 55

¿Cuál es la salida del siguiente fragmento de código?

```
1 | def func(a, b):
2 |     return b ** a
3 |
4 |
5 | print(func(b=2, 2))
```

a) dará como salida 2
b) dará como salida 4
c) es erróneo
d) dará como salida None

Ejercicio 56

¿Qué valor se asignará a la variable x?

```
1 | z = 0
2 | y = 10
3 | x = y < z and z > y or y < z and z < y
4 |
```

a) 1
b) True
c) 0
d) False

Ejercicio 57

¿Cuál es la salida del siguiente fragmento de código?

```
my_list = [x * x for x in range(5)]


def fun(lst):
    del lst[lst[2]]
    return lst


print(fun(my_list))
```

a) [0, 1, 4, 9]
b) [0, 1, 9, 16]
c) [1, 4, 9, 16]
d) [0, 1, 4, 16]

Ejercicio 58

¿Cuál es la salida del siguiente fragmento de código?

```
x = 1
y = 2
x, y, z = x, x, y
z, y, z = x, y, z

print(x, y, z)
```

a) 1 2 1
b) 1 1 2
c) 1 2 2
d) 2 1 2

Ejercicio 59

¿Cuál es la salida del siguiente fragmento de código?

```
a = 1
b = 0
a = a ^ b
b = a ^ b
a = a ^ b

print(a, b)
```

a) 0 0
b) 0 1
c) 1 0
d) 1 1

Ejercicio 60

¿Cuál es la salida del siguiente fragmento de código?

```
def fun(x):
    if x % 2 == 0:
        return 1
    else:
        return 2


print(fun(fun(2)))
```

a) None
b) 1
c) el código provocará un error de ejecución
d) 2

Ejercicio 61

¿Cuál es el resultado del siguiente fragmento de código si el usuario ingresa dos líneas que contienen 3 y 2 respectivamente?

```
x = int(input())
y = int(input())
x = x % y
x = x % y
y = y % x
print(y)
```

a) 1
b) 2
c) 3
d) 0

Ejercicio 62

¿Cuál es el resultado del siguiente fragmento de código si el usuario ingresa dos líneas que contienen 3 y 6 respectivamente?

```
y = input()
x = input()
print(x + y)
```

a) 36
b) 6
c) 3
d) 63

Ejercicio 63

¿Cuál es la salida del siguiente fragmento de código?

```
print("a", "b", "c", sep="sep")

```

a) asepbsepcsep
b) a b c
c) abc
d) asepbsepc

Ejercicio 64

¿Cuál es la salida del siguiente fragmento de código?

```
x = 1 // 5 + 1 / 5
print(x)

```

a) 0
b) 0.5
c) 0.4
d) 0.2

Ejercicio 65

¿Cuál es el resultado del siguiente fragmento de código si el usuario ingresa dos líneas que contienen 2 y 4 respectivamente?

```
x = float(input())
y = float(input())
print(y ** (1 / x))
```

a) 1.0
b) 0.0
c) 2.0
d) 4.2

Ejercicio 66

¿Cuál es la salida del siguiente fragmento de código?

```
dct = {'one': 'two', 'three': 'one', 'two': 'three'}
v = dct['three']

for k in range(len(dct)):
    v = dct[v]

print(v)
```

a) ('one', 'two', 'three')
b) two
c) one
d) three

Ejercicio 67

¿Cuántos elementos contiene la lista lst?

```
lst = [i for i in range(-1, -2)]

```

a) uno
b) cero
c) dos
d) tres

Ejercicio 68

¿Cuáles de las siguientes líneas correctamente invocan la función definida a continuación? (Selecciona dos respuestas)

```
def fun(a, b, c=0):
    # Cuerpo de la función.

```

a) fun(0, 1, 2)
b) fun(b=0, a=0)
c) fun(b=1)
d) fun()

Ejercicio 69

¿Cuál es la salida del siguiente fragmento de código?

```
def fun(x, y):
    if x == y:
        return x
    else:
        return fun(x, y-1)


print(fun(0, 3))
```

a) 1
b) el código provocará un error de ejecución
c) 2
d) 0

Ejercicio 70

¿Cuántos (*) imprimirá el siguiente fragmento de código en la consola?

```
i = 0
while i < i + 2 :
    i += 1
    print("*")
else:
    print("*")
```

a) 1
b) 2
c) el fragmento entrará en un bucle infinito, imprimiendo un * por línea
d) 0

Ejercicio 71

¿Cuál es la salida del siguiente fragmento de código?

```
tup = (1, 2, 4, 8)
tup = tup[-2:-1]
tup = tup[-1]
print(tup)
```

a) 4
b) (4,)
c) 44
d) (4)

Ejercicio 72

¿Cuál es la salida del siguiente fragmento de código?

```
dd = {"1": "0", "0": "1"}
for x in dd.vals():
    print(x, end="")
```

a) 1 0
b) 0 0
c) el código es erróneo (el objeto dict no contiene el método vals())
d) 0 1

Ejercicio 73

¿Cuál es la salida del siguiente fragmento de código?

```
dct = {}
dct['1'] = (1, 2)
dct['2'] = (2, 1)

for x in dct.keys():
    print(dct[x][1], end="")
```

a) 21
b) (2,1)
c) (1,2)
d) 12

Ejercicio 74

¿Cuál es la salida del siguiente fragmento de código?

```
def fun(inp=2, out=3):
    return inp * out


print(fun(out=2))
```

a) 6
b) 4
c) el fragmento de código es erróneo y provocará un SyntaxError
d) 2

Ejercicio 75

¿Cuántos (#) imprimirá el siguiente fragmento de código en la consola?

```
lst = [[x for x in range(3)] for y in range(3)]

for r in range(3):
    for c in range(3):
        if lst[r][c] % 2 != 0:
            print("#")
```

a) tres
b) nueve
c) seis
d) cero

Ejercicio 76

¿Cuál es el comportamiento esperado del siguiente programa si el usuario ingresa 0?

```
1  try:
2      value = input("Ingresa un valor: ")
3      print(int(value)/len(value))
4  except ValueError:
5      print("Entrada incorrecta...")
6  except ZeroDivisionError:
7      print("Entrada erronea...")
8  except TypeError:
9      print("Entrada muy erronea...")
10 except:
11     print("¡Buuu!")
12
```

a) ¡Buuu!
b) 1.0
c) 0.0
d) Entrada muy errónea...
e) Entrada errónea...
f) Entrada incorrecta...

Ejercicio 77

¿Cuál de los siguientes fragmentos muestra la forma correcta de manejar múltiples excepciones en una sola cláusula except?

```python
# A:
except (TypeError, ValueError, ZeroDivisionError):
    # Algo de código.

# B:
except TypeError, ValueError, ZeroDivisionError:
    # Algo de código.

# C:
except: (TypeError, ValueError, ZeroDivisionError)
    # Algo de código.

# D:
except: TypeError, ValueError, ZeroDivisionError
    # Algo de código.

# E:
except (TypeError, ValueError, ZeroDivisionError)
    # Algo de código.

# F:
except TypeError, ValueError, ZeroDivisionError
    # Algo de código.

```

a) A solamente
b) A y B
c) B y C
d) D y E
e) F solamente
f) A, C, y D
g) A y F