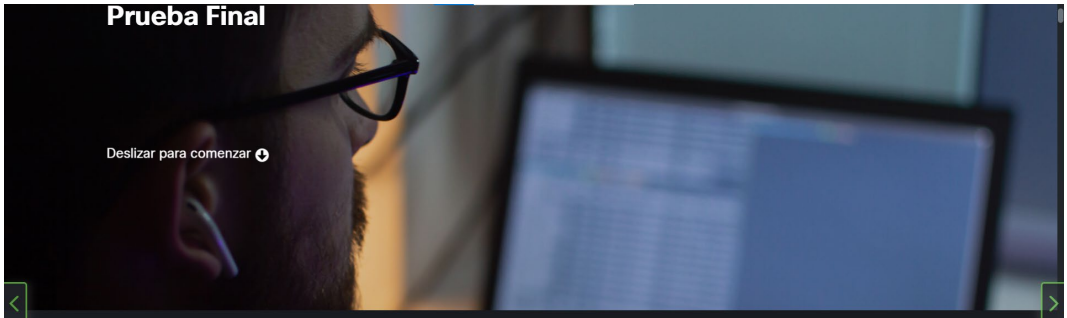
Prueba Final



Deslizar para comenzar

¡Bien hecho! Has llegado al final del curso y completado un paso importante en tu educación de programación en Python.

Ahora estás preparado para enfrentar el último desafío, la **Prueba Final**, que te ayuda a repasar la información más importante que has leído y a evaluar las habilidades y los conocimientos que has adquirido a lo largo del curso.

La **Prueba Final** se basa en lo que has aprendido en todo el curso. Hay 35 preguntas en total, y debes obtener al menos un 70% para aprobar. ¡Buena suerte!

Pregunta 1

¿Cuál es el resultado del siguiente fragmento de código?

```
1  my_list = [1, 2]
2
3  for v in range(2):
4      my_list.insert(-1, my_list[v])
5
6  print(my_list)
7
```

- `[2, 1, 1, 2]`
- `[1, 2, 1, 2]`
- `[1, 2, 2, 2]`
- ✓ `[1, 1, 1, 2]`

Pregunta 2

El significado de un *argumento posicional* está determinado por:

- su valor
- su conexión con variables existentes
- el nombre del argumento especificado junto con su valor
- ✓ su posición dentro de la lista de argumentos

Pregunta 3

¿Cuáles de los siguientes enunciados son **verdaderos** respecto al código? (Selecciona **dos** respuestas)

```
1  nums = [1, 2, 3]
2  vals = nums
3
```

- ✓ `nums` y `vals` son diferentes nombres de la misma lista
- ✓ `nums` tiene la misma longitud que `vals`
- `vals` es más larga que `nums`
- `nums` y `vals` son listas diferentes

Pregunta 4

Un operador capaz de verificar si dos valores **no son iguales** se codifica como:

- `not ==`
- ✓ `!=`
- `<>`
- `=/=`

Pregunta 5

El siguiente fragmento de código:

```
1  def function_1(a):
2      return None
3
4
5  def function_2(a):
6      return function_1(a) * function_1(a)
7
8
9  print(function_2(2))
10
```

- dará como salida `16`
- dará como salida `2`
- ✓ provocará un error de ejecución

Pregunta 6

El resultado de la siguiente división:

```
1 // 2
```

- es igual a `0.5`
- no se puede predecir
- es igual a `0.0`
- ✓ es igual a `0`

Pregunta 7

El siguiente fragmento de código:

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


print(func(b=2, 2))

```

- dará como salida `None`
- dará como salida `4`
- ✓ es erróneo
- dará como salida `2`

Pregunta 8

¿Qué valor se asignará a la variable `x`?

```
z = 0
y = 10
x = y < z and z > y or y < z and z < y

```

- ✓ `False`
- `True`
- `1`
- `0`

Pregunta 9

¿Cuáles de los siguientes nombres de variable son **ilegales** y provocarán una excepción de SyntaxError? (Selecciona **dos** respuestas)

- ✓ `for`
- `print`
- ✓ `in`
- `In`

Pregunta 10

¿Cuál es el resultado del siguiente fragmento de código?

```
1  my_list = [x * x for x in range(5)]
2
3
4  def fun(lst):
5      del lst[lst[2]]
6      return lst
7
8
9  print(fun(my_list))
10
```

- ✓ `[0, 1, 4, 9]`
- `[1, 4, 9, 16]`
- `[0, 1, 4, 16]`

Pregunta 11

¿Cuál es el resultado del siguiente fragmento de código?

```
1  x = 1
2  y = 2
3  x, y, z = x, x, y
4  z, y, z = x, y, z
5
6  print(x, y, z)
7
```

- `1 2 1`
- `2 1 2`
- ✓ `1 1 2`
- `1 2 2`

Pregunta 12

¿Cuál es el resultado del siguiente fragmento de código?

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

print(a, b)

```

- `1 0`
- `1 1`
- ✓ `0 1`
- `0 0`

Pregunta 13

¿Cuál es el resultado del siguiente fragmento de código?

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


print(fun(fun(2)))

```

- `1`
- el código provocará un error de ejecución
- ✓ `2`
- `None`

Pregunta 14

Observa el fragmento de código y elige la sentencia **verdadera**:

```
nums = [1, 2, 3]
vals = nums
del vals[:]

```

- `vals` es más larga que `nums`
- `nums` es más larga que `vals`
- el código provocará un error de ejecución
- ✓ `nums` y `vals` tienen la misma longitud

Pregunta 15

¿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)
```

- `2`
- `1`
- ✓ `0`
- `3`

Pregunta 16

¿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)
```

- ✓ `63`
- `6`
- `36`
- `3`

Pregunta 17

¿Cuál es el resultado del siguiente fragmento de código?

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

- `abc`
- `asepbsepcsep`
- ✓ `asepbsepc`
- `a b c`

Pregunta 18

¿Cuál es el resultado del siguiente fragmento de código?

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

- `0.4`
- `0.5`
- `0`
- ✓ `0.2`

Pregunta 19

Suponiendo que `my_tuple` es una tupla creada correctamente, el hecho de que las tuplas sean inmutables significa que la siguiente instrucción:

```
my_tuple[1] = my_tuple[1] + my_tuple[0]
```

- se puede ejecutar si y solo si la tupla contiene al menos dos elementos
- ✓ es ilegal
- es completamente correcta
- puede ser ilegal si la tupla contiene cadenas

Pregunta 20

¿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))
```

- ✓ `2.0`
- `0.0`
- `1.0`
- `4.2`

Pregunta 21

¿Cuál es el resultado 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)

```

- `two`
- ✓ `one`
- `three`
- `('one', 'two', 'three')`

Pregunta 22

¿Cuántos elementos contiene la lista `lst`?

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

```

- `tres`
- ✓ `cero`
- `dos`
- `uno`

Pregunta 23

¿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.

```

- `fun()`
- ✓ `fun(b=0, a=0)`
- `fun(b=1)`
- ✓ `fun(0, 1, 2)`

Pregunta 24

¿Cuál es el resultado del siguiente fragmento de código?

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


print(fun(0, 3))

```

- 1
- ✓ 0
- 2
- el código provocará un error de ejecución

Pregunta 25

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

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

```

- ✓ el fragmento entrará en un bucle infinito, imprimiendo un * por línea
- 1
- 2
- 0

Pregunta 26

¿Cuál es el resultado del siguiente fragmento de código?

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

```

- 44
- ✓ 4
- (4)
- (4,)

Pregunta 27

¿Cuál es el resultado del siguiente fragmento de código?

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

- `0 1`
- ✓ el código es erróneo (el objeto `dict` no contiene el método `vals()`)
- `1 0`
- `0 0`

Pregunta 28

¿Cuál es el resultado del siguiente fragmento de código?

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

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

- `(1,2)`
- `12`
- `(2,1)`
- ✓ `21`

Pregunta 29

¿Cuál es el resultado del siguiente fragmento de código?

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


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

- ✓ `4`
- `6`
- `2`
- el fragmento de código es erróneo y provocará un SyntaxError

Pregunta 30

¿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("#")

```

- seis
- cero
- nueve
- ✓ tres

Pregunta 31

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

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

```

- `¡Buuu!`
- `Entrada incorrecta...`
- `Entrada muy errónea...`

Pregunta 32

¿Cuál es el comportamiento esperado del siguiente programa?

```
try:
    print(5/0)
    break
except:
    print("Lo siento, algo salió mal...")
except (ValueError, ZeroDivisionError):
    print("Mala suerte...")

```

- ✓ El programa provocará una excepción `SyntaxError`.
- El programa provocará una excepción `ValueError` y dará como salida un mensaje de error predeterminado.
- El programa provocará una excepción `ZeroDivisionError` y dará como salida el siguiente mensaje: `Mala suerte...`
- El programa generará una excepción y será manejada por el primer bloque `except`.

Pregunta 33

¿Cuál es el comportamiento esperado del siguiente programa?

```
foo = (1, 2, 3)
foo.index(0)

```

- El programa provocará una excepción `TypeError`.
- ✓ El programa provocará una excepción `ValueError`.
- El programa provocará una excepción `SyntaxError`.
- El programa dará como salida un `1` en la pantalla.
- El programa provocará una excepción `AttributeError`.

Pregunta 34

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

```
# 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.
```

- D y E
- F solamente
- ✓ A solamente
- A, C, y D
- A y B
- B y C
- A y F

Pregunta 35

¿Qué pasará cuando intentes ejecutar el siguiente código?

```
print(¡Hola, Mundo!)

```

- El código generará la excepción *SyntaxError*.
- El código generará la excepción *ValueError*.
- El código generará la excepción *AttributeError*.
- El código imprimirá `¡Hola, Mundo!` en la consola.
- El código generará la excepción *TypeError*.

Evaluación de revisión

100%

Has obtenido 100%.

Felicidades, has aprobado la evaluación.