PROCESO GESTIÓN DEL TALENTO HUMANO

FORMATO INFORME MENSUAL EJECUCIÓN CONTRACTUAL

Barranquilla, Octubre de 2022

Señor (a)
MANUEL GREGORIO HORMECHEA LANCE
SUPERVISOR(A) CONTRATO No. **3298199**
Coordinador Sede TIC
TIC Y Energía
Barranquilla

Asunto: Informe mensual de ejecución contractual
Mes Octubre del año 2022

Referencia: No 3298199 del año 2022

LILIOLA PAOLA PRINS BALLESTAS, identificado con la cédula de ciudadanía No. 32.780.488 de Barranquilla, en mi calidad de Contratista del SENA, en TIC Y Energía, en cumplimiento del Contrato de Prestación de Servicios de la referencia, a continuación presento el Informe de actividades realizadas en el mes objeto de cobro.

Valor y forma de Pago: j). un pago correspondiente al mes de Octubre de 2022 por valor de (3.477.822). Los honorarios serán pagados por el SENA al contratista de acuerdo al cronograma definido por la Dirección Administrativa y Financiera de la Dirección General, en la cuenta de Ahorros No. 620280107 BBVA, cuyo titular es el Contratista.

Plazo: Será hasta el 17 de Diciembre de 2022.

OBJETO:
Prestar los servicios profesionales como Instructor para impartir formación en Programación de Software de acuerdo con la planeación de la coordinación de TIC y Energía del Centro Nacional Colombo Alemán

Obligaciones Especificas: *(Trascriba las obligaciones especificas del contrato, dentro del siguiente cuadro)*

No	Obligaciones	Acciones realizadas	Evidencias
1	Impartir formación a las fichas asignadas	Seleccionar las estrategias de enseñanza, aprendizaje y evaluación pertinentes para el grupo. Orientar el proceso de aprendizaje de acuerdo con el diseño curricular vigente. Reportar la información académica y administrativa que se generen en el proceso.	Se realizó solicitud de creación de las fichas.

2	Impartir formación a las fichas asignadas	Sesiones de orientación mediadas por tecnología (Google Meeting - WhatsApp)	Capturas de pantalla tomadas de las sesiones
3	Impartir formación a las fichas asignadas	Espacios Virtuales disponibles en la plataforma institucional	Materiales y proyectos publicados en la plataforma
4	Impartir formación a las fichas asignadas	Preparación BootCamp como estrategia de aprehendizaje	Materiales diseñados, bajo el concepto de intencionalidad académica
5	Coordinar estrategias ciudadanía digital	Socializar y apoyar a los aprendices en la inscripción y realización de los cursos para acreditarse como ciudadano digital	Reuniones, listas de certificaciones, listas de inscritos.

A continuación relaciono los desplazamientos que realicé previo a la presentación de este informe. Una vez finalizado cada desplazamiento presenté al ordenador del gasto el informe en el Formato Informe Legalización Desplazamiento Contratista GTH-F-087, en el que se describieron las actividades desarrolladas y los resultados de cada desplazamiento. Cada informe cuenta con el visto bueno del Supervisor.

Se lista a continuación el soporte de la legalización de los desplazamientos realizados, los cuales forman parte integral del presente informe de ejecución contractual.

ITEM	No DE LA ORDEN DE VIAJE	LUGAR DE DESPLAZAMIENTO	FECHAS DE DESPLAZAMIENTOS	
			INICIO	FINALIZACIÓN

Nota 1: *Por cada desplazamiento que haya realizado el contratista, adjuntará el respectivo informe que la soporte. En caso de haber realizado el desplazamiento en fecha posterior a la presentación del informe de ejecución contractual, deberá reportarlo en el siguiente informe de ejecución contractual.*

Para el trámite de la cuenta me permito adjuntar: Documentos electrónicos enunciados como evidencias del cumplimiento de las obligaciones contractuales y los desplazamientos realizados y el No. 7852535557 de la planilla, operador y periodo. (Decreto Ley 2106 de 2019 – " *Decreto Ley Antitrámites"*)

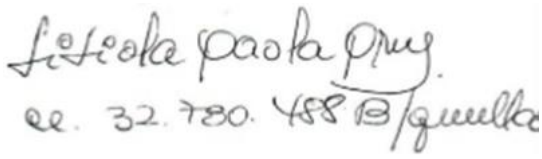
Declaración bajo la gravedad de juramento.

Me permito declarar **bajo la gravedad de juramento** que durante el periodo objeto de cobro SI____NO_X_ he suscrito otro u otros contratos con entidade(s) del Estado el (los) cual(es) se encuentra(n) en ejecución; la anterior declaración de acuerdo con lo establecido en el numeral 5º de la Directiva Presidencial No. 001 de 2022, a fin de identificar potenciales conflictos de interés y hacer la gestión del riesgos para la toma de decisiones dentro del principio de la debida diligencia frente a la ejecución de los contratos de prestación de servicios profesionales o de apoyo a la gestión.

Nota: En el evento de señalar **SI,** el análisis efectuado por el supervisor y ordenador de pago deberá quedar documentado y publicado en el SECOP II en los documentos relacionados en el archivo Gestión Contractual GC_ del periodo objeto de cobro.

Evidencias en () folios

Cordialmente,



LILIOLA PAOLA PRINS BALLESTAS
Contratista
C.C. No. 32.780.488

Recibí a satisfacción:

MANUEL GREGORIO HORMECHEA LANCE
Supervisor(a) Contrato 3298199 de 2022
Coordinador Sede TIC

EVIDENCIAS

Manejo de listas, generación de números y entradas de usuario.

```
# -*- coding: utf-8 -*-

from random import randint
import os
import time

class Jugador():
    def
__init__(self,nombre,tipo,guerra,oceano,jugadas,barco
s):
        self.nombre = nombre
        self.tipo = tipo
        self.guerra = guerra
        self.oceano = oceano
        self.jugadas = jugadas
        self.barcos = barcos

    def _actualizar(self):
        self.barcos=self.barcos-1

    def _contar(self,X,Y,V):
        C = 0
        # Primero intenta contar a la izquierda
        while X>1 and self.guerra.tablero[X][Y]==V:
            C = C+1
```

```
            X = X-1
        # Ahora intenta contar a la derecha
        while X<11 and self.guerra.tablero[X][Y]==V:
            C = C+1
            X = X+1
        # Ahora intenta contar hacia arriba
        while Y>1 and self.guerra.tablero[X][Y]==V:
            C = C+1
            Y = Y-1
        # Ahora intenta contar hacia abajo
        while Y<11 and self.guerra.tablero[X][Y]==V:
            C = C+1
            Y = Y+1
        return C

    def _automatica(self):
        vale=False
        while not vale:
            X = randint(1,10)
            Y = randint(1,10)
            jugada = Jugada(X,Y)
            if not jugada in self.jugadas:
                vale = True
                return jugada

    def _manual(self):
        X = 0
```

```
        Y = 0
        while X<1 or X>10:
            X = read_int("Digite la fila para su
jugada (un valor entre 1 y 10)    : ")
        while Y<1 or Y>10:
            Y = read_int("Digite la columna para su
jugada (un valor entre 1 y 10) : ")
        jugada = Jugada(X,Y)
        return jugada

class Jugada():
    def __init__(self,X,Y):
        self.X = X
        self.Y = Y

class Tablero():
    def __init__(self):
        self.tablero = [
                        ["  ", "  1", "  2", "  3",
"  4", "  5", "  6", "  7", "  8", "  9", " 10"],
                        [" A", "    ", "    ", "    ",
"    ", "    ", "    ", "    ", "    ", "    "],
                        [" B", "    ", "    ", "    ",
"    ", "    ", "    ", "    ", "    ", "    "],
                        [" C", "    ", "    ", "    ",
"    ", "    ", "    ", "    ", "    ", "    "],
```

```
                        [" D", "   ", "   ", "   ",
"   ", "   ", "   ", "   ", "   ", "   ", "   ", "   "],
                        [" E", "   ", "   ", "   ",
"   ", "   ", "   ", "   ", "   ", "   ", "   ", "   "],
                        [" F", "   ", "   ", "   ",
"   ", "   ", "   ", "   ", "   ", "   ", "   ", "   "],
                        [" G", "   ", "   ", "   ",
"   ", "   ", "   ", "   ", "   ", "   ", "   ", "   "],
                        [" H", "   ", "   ", "   ",
"   ", "   ", "   ", "   ", "   ", "   ", "   ", "   "],
                        [" I", "   ", "   ", "   ",
"   ", "   ", "   ", "   ", "   ", "   ", "   ", "   "],
                        [" J", "   ", "   ", "   ",
"   ", "   ", "   ", "   ", "   ", "   ", "   ", "   "]
                    ]

    def _print(self):
        for fil in self.tablero:
            for col in fil:
                print(col, end=' | ')
            print()
            print('   ------------------------------
------------------------------')

    def _buscar(self,X,Y,D,N):
        C = 0
        Sw = True
```

```
        while C<N and X<11 and Y<11 and Sw:
            if(self.tablero[X][Y]=="   "):
                C = C+1
                if D==1:
                    Y = Y+1
                else:
                    X = X+1
            else:
                Sw = False
        if C>=N:
            return True
        else:
            return False

    def _ubicar(self,X,Y,D,N,V):
        for i in range(0,N):
            self._marcar(X,Y,V)
            if D==1:
                Y = Y+1
            else:
                X = X+1

    def _marcar(self,X,Y,V):
        self.tablero[X][Y] = V

    def _automatico(self):
```

```
        # Esta rutina posiciona los barcos de forma
automática, tener en cuenta que:
        # Un (1) Porta Aviones --> Ocupa 5 casillas
        # Un (1) Buque         --> Ocupa 4 Casillas
        # Dos (2) Submarinos   --> Ocupan 3 casillas
        # Tres (3) Lanchas     --> Ocupan 2 casillas
        # Primero se ubica el Porta Aviones
        vale=False
        while not vale:
            X = randint(1,10)
            Y = randint(1,10)
            D = randint(1,2) # 1 --> Horizontal, 2
Vertical
            vale = self._buscar(X,Y,D,5)
            if vale:
                self._ubicar(X,Y,D,5,' P ')
        # Se ubica el Buque
        vale=False
        while not vale:
            X = randint(1,10)
            Y = randint(1,10)
            D = randint(1,2) # 1 --> Horizontal, 2
Vertical
            vale = self._buscar(X,Y,D,4)
            if vale:
                self._ubicar(X,Y,D,4,' B ')
        # Se ubica el primer Submarino
```

```
        vale=False
        while not vale:
            X = randint(1,10)
            Y = randint(1,10)
            D = randint(1,2) # 1 --> Horizontal, 2
Vertical
            vale = self._buscar(X,Y,D,3)
            if vale:
                self._ubicar(X,Y,D,3,' S1')
        # Se ubica el segundo Submarino
        vale=False
        while not vale:
            X = randint(1,10)
            Y = randint(1,10)
            D = randint(1,2) # 1 --> Horizontal, 2
Vertical
            vale = self._buscar(X,Y,D,3)
            if vale:
                self._ubicar(X,Y,D,3,' S2')
        # Se ubica la primera Lancha
        vale=False
        while not vale:
            X = randint(1,10)
            Y = randint(1,10)
            D = randint(1,2) # 1 --> Horizontal, 2
Vertical
            vale = self._buscar(X,Y,D,2)
```

```
            if vale:
                self._ubicar(X,Y,D,2,' L1')
        # Se ubica la segunda Lancha
        vale=False
        while not vale:
            X = randint(1,10)
            Y = randint(1,10)
            D = randint(1,2) # 1 --> Horizontal, 2
Vertical
            vale = self._buscar(X,Y,D,2)
            if vale:
                self._ubicar(X,Y,D,2,' L2')
        # Se ubica la tercera Lancha
        vale=False
        while not vale:
            X = randint(1,10)
            Y = randint(1,10)
            D = randint(1,2) # 1 --> Horizontal, 2
Vertical
            vale = self._buscar(X,Y,D,2)
            if vale:
                self._ubicar(X,Y,D,2,' L3')

    def _manual(self):
        # Esta rutina posiciona los barcos de forma
manual, tener en cuenta que:
        # Un (1) Porta Aviones --> Ocupa 5 casillas
```

```
        # Un (1) Buque          --> Ocupa 4 Casillas
        # Dos (2) Submarinos    --> Ocupan 3 casillas
        # Tres (3) Lanchas      --> Ocupan 2 casillas
        # Primero se ubica el Porta Aviones
        vale=False
        while not vale:
            X = 0
            Y = 0
            while X<1 or X>10:
                X = read_int("Digite la fila en la
que ubicará el Porta Aviones (un valor entre 1 y
10)    : ")
            while Y<1 or Y>10:
                Y = read_int("Digite la columna en la
que ubicará el Porta Aviones (un valor entre 1 y 10)
: ")
            D = int(read_Range("En que sentido
ubicará el barco 1) Horizontal o 2)
Vertical                    : ",["1","2"]))
            vale = self._buscar(X,Y,D,5)
            if vale:
                self._ubicar(X,Y,D,5,' P ')
            else:
                print("\n\tNo se puede ubicar el
Porta Aviones en esa posición, por favor reintente")
        # Se ubica el Buque
        vale=False
```

```
        while not vale:
            X = 0
            Y = 0
            while X<1 or X>10:
                X = read_int("Digite la fila en la
que ubicará el Buque (un valor entre 1 y
10)            : ")
            while Y<1 or Y>10:
                Y = read_int("Digite la columna en la
que ubicará el Buque (un valor entre 1 y
10)          : ")
            D = int(read_Range("En que sentido
ubicará el barco 1) Horizontal o 2)
Vertical                    : ",["1","2"]))
            vale = self._buscar(X,Y,D,4)
            if vale:
                self._ubicar(X,Y,D,4,' B ')
            else:
                print("\n\tNo se puede ubicar el
Buque en esa posición, por favor reintente")
        # Se ubica el primer Submarino
        vale=False
        while not vale:
            X = 0
            Y = 0
            while X<1 or X>10:
```

```
                X = read_int("Digite la fila en la
que ubicará el Submarino 1 (un valor entre 1 y
10)      : ")
            while Y<1 or Y>10:
                Y = read_int("Digite la columna en la
que ubicará el Submarino 1 (un valor entre 1 y
10)   : ")
            D = int(read_Range("En que sentido
ubicará el barco 1) Horizontal o 2)
Vertical                    : ",["1","2"]))
            vale = self._buscar(X,Y,D,3)
            if vale:
                self._ubicar(X,Y,D,3,' S1')
            else:
                print("\n\tNo se puede ubicar el
Submarino 1 en esa posición, por favor reintente")
        # Se ubica el segundo Submarino
        vale=False
        while not vale:
            X = 0
            Y = 0
            while X<1 or X>10:
                X = read_int("Digite la fila en la
que ubicará el Submarino 2 (un valor entre 1 y
10)      : ")
            while Y<1 or Y>10:
```

```
                Y = read_int("Digite la columna en la
que ubicará el Submarino 2 (un valor entre 1 y
10)   : ")
            D = int(read_Range("En que sentido
ubicará el barco 1) Horizontal o 2)
Vertical                    : ",["1","2"]))
            vale = self._buscar(X,Y,D,3)
            if vale:
                self._ubicar(X,Y,D,3,' S2')
            else:
                print("\n\tNo se puede ubicar el
Submarino 2 en esa posición, por favor reintente")
        # Se ubica la primera Lancha
        vale=False
        while not vale:
            X = 0
            Y = 0
            while X<1 or X>10:
                X = read_int("Digite la fila en la
que ubicará La Lancha 1 (un valor entre 1 y
10)         : ")
            while Y<1 or Y>10:
                Y = read_int("Digite la columna en la
que ubicará La Lancha 1 (un valor entre 1 y
10)      : ")
```

```
            D = int(read_Range("En que sentido
ubicará el barco 1) Horizontal o 2)
Vertical                    : ",["1","2"]))
            vale = self._buscar(X,Y,D,2)
            if vale:
                self._ubicar(X,Y,D,2,' L1')
            else:
                print("\n\tNo se puede ubicar la
Lancha 1 en esa posición, por favor reintente")
        # Se ubica la segunda Lancha
        vale=False
        while not vale:
            X = 0
            Y = 0
            while X<1 or X>10:
                X = read_int("Digite la fila en la
que ubicará La Lancha 2 (un valor entre 1 y
10)         : ")
            while Y<1 or Y>10:
                Y = read_int("Digite la columna en la
que ubicará La Lancha 2 (un valor entre 1 y
10)      : ")
            D = int(read_Range("En que sentido
ubicará el barco 1) Horizontal o 2)
Vertical                    : ",["1","2"]))
            vale = self._buscar(X,Y,D,2)
            if vale:
```

```
                self._ubicar(X,Y,D,2,' L2')
            else:
                print("\n\tNo se puede ubicar la
Lancha 2 en esa posición, por favor reintente")
        # Se ubica la tercera Lancha
        vale=False
        while not vale:
            X = 0
            Y = 0
            while X<1 or X>10:
                X = read_int("Digite la fila en la
que ubicará La Lancha 3 (un valor entre 1 y
10)         : ")
            while Y<1 or Y>10:
                Y = read_int("Digite la columna en la
que ubicará La Lancha 3 (un valor entre 1 y
10)      : ")
            D = int(read_Range("En que sentido
ubicará el barco 1) Horizontal o 2)
Vertical                    : ",["1","2"]))
            vale = self._buscar(X,Y,D,2)
            if vale:
                self._ubicar(X,Y,D,2,' L3')
            else:
                print("\n\tNo se puede ubicar la
Lancha 3 en esa posición, por favor reintente")
```

```
def read_int(msg):
    try:
      return int(input(msg))
    except:
      return read_int('\n\tEl dato introducido no es
correcto, vuelva a intentarlo : ')

def read_Range(msg,rng):
    dat = input(msg)
    if dat in rng:
      return dat
    else:
      return read_Range('\n\tEl dato introducido no
es correcto, vuelva a intentarlo : ',rng)

def Jugar(J1, J2):
    # se determina quien inicia (con un numero
Aleatorio)
    # Tener en cuenta que los jugadores humanos
juegan de forma manual, la máquina juega de forma
automática
    # La máquina nunca perderá turnos por repetir una
jugada, el humano puede perderlos por esa razón
    I = randint(1,2)
    if (I==1):
        Juega = J1
        Otro = J2
```

```
    else:
        Juega = J2
        Otro = J1
    # El juego termina cuando alguno de los jugadores
destruya todos los barcos del enemigo
    Fin = False
    Turno = 0
    while not Fin:
        Turno = Turno + 1
        os.system("cls")
        if Juega.tipo=="M":
            Jugada = Juega._automatica()
        else:
            Jugada = Juega._manual()
        print("\n\t --> Es el turno "+str(Turno)+",
Juega "+Juega.nombre+" ::: Posición :
["+str(Jugada.X)+" - "+str(Jugada.Y)+"]")
        if Jugada in Juega.jugadas:
            print("\n\t\t --> Movimiento invalido,
pierdes el turno ")
            if Juega== J1:
               Juega = J2
               Otro = J1
            else:
                Juega = J1
                Otro = J2
        else:
```

```
            Juega.jugadas.append(Jugada)
            # Se determina el resultado de la jugada
sobre el tablero del contrario
            X = Jugada.X
            Y = Jugada.Y
            res = Otro.guerra.tablero[X][Y]
            if res == "   ": # falla y pierde el
turno
                print("\n\t\t --> ¡Falla!, cedes el
turno")
                Juega.oceano.tablero[X][Y] = " F "
                if Juega== J1:
                    Juega = J2
                    Otro = J1
                else:
                    Juega = J1
                    Otro = J2
            else:   # Se determina si el barco fue
¡tocado! o ¡hundido!
                Juega.oceano.tablero[X][Y] = " H "
                res = "*"+res[1:] #Inicialmente se
marca como ¡tocado!
                Otro.guerra.tablero[X][Y] = res
                con = Otro._contar(X,Y,res)
                if res=="*P ":
                    if con==5:
                        msg = "¡Hundido!"
```

```
                    else:
                        msg = "¡Tocado!"
                elif res=="*B ":
                    if con==4:
                        msg = "¡Hundido!"
                    else:
                        msg = "¡Tocado!"
                elif res=="*S1" or res=="*S2":
                    if con==3:
                        msg = "¡Hundido!"
                    else:
                        msg = "¡Tocado!"
                else:
                    if con==2:
                        msg = "¡Hundido!"
                    else:
                        msg = "¡Tocado!"
                print("\t\t\t ===>>Un barco de
"+Otro.nombre+", ha sido "+msg)
                if(msg=="¡Hundido!"):
                    Otro._actualizar()
                    if Otro.barcos<1:
                       print("\t\t\t ::::::::: Juego
terminado, ha ganado "+Juega.nombre)
                       Fin = True
        Juega.oceano._print()
        time.sleep(2)
```

```
def logo():
    print("0110001   110000  01110100 01110100
01       01100101  1110011 01    00 01101001
0111000")
    print("01    10
01    01    10       10    01       01       01
01    00    01    01    00")
    print("01    10
01    01    10       10    01       01       01
01    00    01    01    00")
    print("01100010
01100001    10       10    01       01100101  111001
 01101000    01    0111000")
    print("01    10
01    01    10       10    01       01            11
01    00    01    01")
    print("01    10
01    01    10       10    01       01            11
01    00    01    01")
    print("0110001  01    01    10       10    011011
00 01100101 0111001  01    00 01101001 01")

def P2P():
    # se crea el primer jugador (Humano)
    guerra1 = Tablero()
    oceano1 = Tablero()
```

```
    nom=input("Por favor digite el nombre para el
primer jugador                          : ")
    op = read_Range("El jugador 1 desea ubicar sus
barcos de forma A)utomática o M)anual :
",["A","a","M","m"])
    if op=="A" or op=="a":
        guerra1._automatico()
    else:
        guerra1._manual()
    Jugador1 =  Jugador(nom,"H",guerra1, oceano1, [],
7)
    # Se crea el segundo jugador (Humano)
    guerra2 = Tablero()
    oceano2 = Tablero()
    nom=input("Por favor digite el nombre para el
segundo jugador                         : ")
    op = read_Range("El jugador 2 desea ubicar sus
barcos de forma A)utomática o M)anual :
",["A","a","M","m"])
    if op=="A" or op=="a":
        guerra2._automatico()
    else:
        guerra2._manual()
    Jugador2 =  Jugador(nom,"H",guerra2, oceano2, [],
7)
    Jugar(Jugador1, Jugador2)
```



```
def P2B():
    # se crea el primer jugador (Humano)
    guerra1 = Tablero()
    oceano1 = Tablero()
    nom=input("Por favor digite el nombre para el
jugador                                    : ")
    op = read_Range("El jugador 1 desea ubicar sus
barcos de forma A)utomática o M)anual  :
",["A","a","M","m"])
    if op=="A" or op=="a":
        guerra1._automatico()
    else:
        guerra1._manual()
    Jugador1 =  Jugador(nom,"H",guerra1, oceano1, [],
7)
    # Se crea el segundo jugador (Máquina)
    guerra2 = Tablero()
    oceano2 = Tablero()
    nom="Isaak Yúdovich Azímov"
    guerra2._automatico()
    Jugador2 =  Jugador(nom,"M",guerra2, oceano2, [],
7)
    Jugar(Jugador1, Jugador2)

def B2B():
    # Se crea el primer jugador (Máquina)
    guerra1 = Tablero()
```

```
    oceano1 = Tablero()
    nom="Leonardo Di Ser Piero Da Vinci"
    guerra1._automatico()
    Jugador1 =  Jugador(nom,"M",guerra1, oceano1, [],
7)
    # Se crea el segundo jugador (Máquina)
    guerra2 = Tablero()
    oceano2 = Tablero()
    nom="Isaak Yúdovich Azímov"
    guerra2._automatico()
    Jugador2 =  Jugador(nom,"M",guerra2, oceano2, [],
7)
    Jugar(Jugador1, Jugador2)

def ShowRules():
    print("Tableros")
    print()
    print("\nLos jugadores manejan un tablero de
océano y un tablero de tiro; cada uno dividido en
casillas. Cada tablero representa una zona diferente
del mar abierto: la propia y la contraria. En el
primer tablero, el jugador coloca sus barcos y
registra los «tiros» del oponente; en el otro, se
registran los tiros propios contra el otro jugador,
diferenciando los impactos y los que dan al agua.")
    print()
    print("\nBarcos")
```

```
    print()
    print("\nAl comenzar, cada jugador posiciona sus
barcos en el primer tablero, de forma secreta,
invisible al oponente.")
    print("\nCada quien ocupa, según sus
preferencias, una misma cantidad de casillas,
horizontal y/o verticalmente, las que representan sus
naves. Ambos participantes deben ubicar igual número
de naves, por lo que es habitual, antes de comenzar,
estipular de común acuerdo la cantidad y el tamaño de
las naves que se posicionarán en el tablero. Así, por
ejemplo, cinco casillas consecutivas conforman un
portaaviones; cuatro, un buque; tres, un submarino; y
dos, una lancha. En esta versión del juego cada
jugador dispone de Un (1) Porta Aviones, Un (1)
Buque, Dos (2) Submarinos y Tres (3) Lanchas")
    print()
    print("\nMecánica del juego")
    print()
    print("Una vez todas las naves han sido
posicionadas, se inicia una serie de rondas. En cada
ronda, cada jugador en su turno «dispara» hacia la
flota de su oponente indicando una posición (las
coordenadas de una casilla), la que registra en el
segundo tablero. Si esa posición es ocupada por parte
de un barco contrario, el oponente indicará ¡Tocado!,
si todavía quedan partes del barco (casillas) sin
```

```
dañar, o ¡Hundido! si con ese disparo la nave ha
quedado totalmente destruida (esto es, si la acertada
es la última de las casillas que conforman la nave
que quedaba por acertar). El jugador que ha tocado un
barco en su anterior jugada, volverá a disparar hasta
que falle. Si la posición indicada no corresponde a
una parte de barco alguno, cantará ¡Falla! y el turno
pasa al otro jugador")
    print()
    print("Fin del juego")
    print()
    print("El juego termina cuando alguno de los
jugadores destruye todas las naves de su oponente.
Este jugador se declara el vencedor")

def main():
    op = "N"
    while op!="S" and op!="s":
        os.system("cls")
        logo()
        print("\t\nBIENVENIDO AL JUEGO BATALLA
NAVAL\n")
        print("\tOPCIONES DISPONIBLES\n")
        print("\t1. Reglas del juego")
        print("\t2. Jugador Vs Jugador")
        print("\t3. Jugador Vs Bot")
        print("\t4. Bot Vs Bot")
```

```
        print("\tS. Salir")
        op = read_Range("\n\tSeleccione su opción :
",['1','2','3','4','S','s'])
        if op=="1":
            ShowRules()
        elif op=="2":
            P2P()
        elif op=="3":
            P2B()
        elif op=="4":
            B2B()
        else:
            print("\tHasta la vista ...")
        input("Enter para continuar")

main()
```

```
from tkinter import Tk, Button, messagebox

symbol = 'O'
board = [['','',''],['','',''],['','','']]

def gamestatus(status=False):
    global symbol
    global board
    if not status:
        btn00['state'] = 'disabled'
        btn01['state'] = 'disabled'
        btn02['state'] = 'disabled'
        btn10['state'] = 'disabled'
        btn11['state'] = 'disabled'
        btn12['state'] = 'disabled'
        btn20['state'] = 'disabled'
        btn21['state'] = 'disabled'
        btn22['state'] = 'disabled'
    else:
        btn00['state'] = 'normal'
        btn01['state'] = 'normal'
        btn02['state'] = 'normal'
        btn10['state'] = 'normal'
        btn11['state'] = 'normal'
        btn12['state'] = 'normal'
        btn20['state'] = 'normal'
```

```
        btn21['state'] = 'normal'
        btn22['state'] = 'normal'
        btn00['text'] = ''
        btn01['text'] = ''
        btn02['text'] = ''
        btn10['text'] = ''
        btn11['text'] = ''
        btn12['text'] = ''
        btn20['text'] = ''
        btn21['text'] = ''
        btn22['text'] = ''
        symbol = 'O'
        board = [['','',''],['','',''],['','','']]

def checkresult(fil,col):
    global board
    dato = board[fil][col]
    sw   = True
    for tmp in range(3):                # Verifico si
se completó una fila
        if board[fil][tmp]!=dato:
            sw = False
            break
    if not sw:
        sw = True
        for tmp in range(3):            # Verifico si
se completó una columna
```

```
            if board[tmp][col]!=dato:
                sw = False
                break
    if not sw and fil==col:            # Es un
elemento de la diagonal principal
        sw = True
        for tmp in range(3):
            if board[tmp][tmp]!=dato:
                sw = False
                break
    if not sw and fil+col==2:          # Es un
elemento de la diagonal secundaria
        sw = True
        for tmp in range(3):
            if board[tmp][2-tmp]!=dato:
                sw = False
                break
    if sw:
        if board[fil][col]=='X':
            res = "Player 2 WON"
        else:
            res = "Player 1 WON"
        gamestatus(False)
    else:
        sw = True
        for ele in board:
            for dat in ele:
```

```
                if dat=='':
                    sw = False
                    break
        if sw:
            res = "Draw Game"
        else:
            res = ''
    return res

def putmark(btn,fil,col):
    global symbol
    global board
    btn['text'] = symbol
    board[fil][col] = symbol
    res=checkresult(fil,col)
    if res!='':
        messagebox.showinfo(message=res, title="Game
results")
    else:
        if symbol=='X':
            symbol ='O'
        else:
            symbol = 'X'

def btn00clk():
    putmark(btn00,0,0)
    btn00['state'] = 'disabled'
```



```
def btn01clk():
    putmark(btn01,0,1)
    btn01['state'] = 'disabled'

def btn02clk():
    putmark(btn02,0,2)
    btn02['state'] = 'disabled'

def btn10clk():
    putmark(btn10,1,0)
    btn10['state'] = 'disabled'

def btn11clk():
    putmark(btn11,1,1)
    btn11['state'] = 'disabled'

def btn12clk():
    putmark(btn12,1,2)
    btn12['state'] = 'disabled'

def btn20clk():
    putmark(btn20,2,0)
    btn20['state'] = 'disabled'

def btn21clk():
    putmark(btn21,2,1)
```



```
    btn21['state'] = 'disabled'

def btn22clk():
    putmark(btn22,2,2)
    btn22['state'] = 'disabled'

def btn30clk():
    gamestatus(True)

window = Tk()
btn00 = Button(window, text='', fg='blue', height=5,
width=10, command=btn00clk)
btn01 = Button(window, text='', fg='blue', height=5,
width=10, command=btn01clk)
btn02 = Button(window, text='', fg='blue', height=5,
width=10, command=btn02clk)
btn10 = Button(window, text='', fg='blue', height=5,
width=10, command=btn10clk)
btn11 = Button(window, text='', fg='blue', height=5,
width=10, command=btn11clk)
btn12 = Button(window, text='', fg='blue', height=5,
width=10, command=btn12clk)
btn20 = Button(window, text='', fg='blue', height=5,
width=10, command=btn20clk)
btn21 = Button(window, text='', fg='blue', height=5,
width=10, command=btn21clk)
```

```
btn22 = Button(window, text='', fg='blue', height=5,
width=10, command=btn22clk)
btn30 = Button(window, text='New game', fg='blue',
height=3, width=15, command=btn30clk)
#btn22.bind('<Button-1>',btn22clk)
btn00.place(x=25,y=20)
btn01.place(x=105,y=20)
btn02.place(x=185,y=20)
btn10.place(x=25,y=106)
btn11.place(x=105,y=106)
btn12.place(x=185,y=106)
btn20.place(x=25,y=192)
btn21.place(x=105,y=192)
btn22.place(x=185,y=192)
btn30.place(x=150,y=280)
window.title('TicTacToe V1.0')
# Se usa para centrar la ventana
vw = 290
vh = 350
vx = window.winfo_screenwidth() // 2 - vw // 2
vy = window.winfo_screenheight() // 2 - vh // 2
posicion = str(vw) + "x" + str(vh) + "+" + str(vx) +
"+" + str(vy)
window.geometry(posicion)
window.resizable(0,0)
window.mainloop()
```

```
'''
Este programa representa el juego de adivinar un
número en n intentos; para lo cual se:
    a) Selecciona aleatoriamente (piensa) un número
de 4 cifras.
    b) Se permite un máximo de 15 intentos
    c) Por cada uno de los intentos, el jugador
indica un número y:
        1) Si el número "pensado" por el computador
es mayor se indica mayor
        2) Si el número "pensado" por el computador
es menor se indica menor
        3) Si los números coinciden se indica que
ganó.

ANALISIS
    META        :   Diseñar el programa que permita
jugar

    ENTRADAS    :   num     :   número ingresado por
el jugador para intentar adivinar

    SALIDAS     :   msg     :   mensaje indicando si
el jugador logró o no adivinar el
                                número
```

```
    VARIABLES   :   num_pen :   Número seleccionado
(pensado) por el computador
                    num_int :   Número máximo de
intentos para adivinar el número
                    intento :   En que intento va el
jugador

    PROCESO     :   a)  El computador (programa)
genera un número
                    b)  Se inicializan todas las
variables de proceso
                    c)  En un ciclo repetitivo que se
ejecuta hasta que se adivine el
                        número o hasta que se agoten
los intentos
                        0)  Se incrementa el número
de intento
                        1)  El jugador indica un
número
                        2)  El programa informa mas
alto, mas bajo o ganaste
    ALGORITMO
        Inicio
            num_pen <- aleatorioentre(1000,9999)
            num_int <- 15
            intento <- 1
            num <- 0
```

```
            MQ intento<num_int y num<>num_pen Haga
                lea num
                Si num<>num_pen Entonces
                    Si num<num_pen Entonces
                        Escriba "mas alto"
                    Sino
                        Escriba "mas bajo"
                    FinSi
                    intento <- intento + 1
                FinSi
            FMQ
            Si num=num_pen Entonces
                Escriba "Felicitaciones, lo lograste
en ",intento," intentos"
            Sino
                Escriba "Lo siento, se agotaron los
turnos. El número a adivinar era ",num_pen
            FinSi
        Fin

    COMENTARIOS
        La versión implementada en pyton, introduce
validación de entradas para evitar
        fallos del programa al momento que el usuario
introduce un valor.

    RESULTADOS DE UN CASO DE PRUEBA.
```

```
        Intento  1 : 1
        Por favor indique un número entero válido
entre 1000 y 9999
        Intento  1 : 5000
        Mas alto
        Intento  2 : 7500
        Mas alto
        Intento  3 : 8700
        Mas bajo
        Intento  4 : 8000
        Mas alto
        Intento  5 : 8350
        Mas bajo
        Intento  6 : 8200
        Mas bajo
        Intento  7 : 8100
        Mas alto
        Intento  8 : 8150
        Mas bajo
        Intento  9 : 8125
        Mas alto
        Intento 10 : 8140
        Mas alto
        Intento 11 : 8145
        Mas bajo
        Intento 12 : 8142
        Mas alto
```

```
        Intento 13 : 8144
        Mas bajo
        Intento 14 : 8143
        Felicidades lo lograste en 14 intentos
'''
from random import randint    ## Se selecciona el
módulo para generar (pensar) el número
def entradaentera(msg='Digite un número entero',
men=0, may=9999999999):
    while True:
        try:
            num = int(input(msg))
            if men<=num<=may:
                break
            else:
                print(f'Por favor indique un número
entero válido entre {men} y {may}')
        except Exception:
            print(f'Por favor indique un número
entero válido entre {men} y {may}')
    return num

num_pen = randint(1000,9999)  ## Se genera (piensa),
el número
num_int = 15                  ## Se establece el
número máximo de intentos
```

```
intento = numero = 0          ## No se ha realizado
el primer intento
while intento<num_int and numero!=num_pen:
    intento += 1
    numero = entradaentera(f'Intento {intento:2d} :
',1000, 9999)
    if numero<num_pen:
        print("Mas alto")
    elif numero>num_pen:
        print("Mas bajo")
if numero==num_pen:
    print(f'Felicidades lo lograste en {intento}
intentos')
else:
    print(f'Sigue intentando, el número que pensé fue
:: {num_pen}')

##
## NOTA: Aplicar las modificaciones necesarias para
que el programa permita jugar mas de
# una vez
##
```