Python como primer lenguaje de programación: un enfoque orientado a Juegos

Elizabeth Vidal Duarte[1]

evidal@ulasalle.edu.pe

Carrera Profesional de Ingeniería del Software
Universidad La Salle, Perú
Av. Alfonso Ugarte 517
Arequipa – Perú

Resumen: *En este artículo, se describe nuestra experiencia en la utilización de Python como primer lenguaje de programación. Se diseñaron una secuencia de laboratorios en donde, mediante un enfoque orientado a la creación de juegos, los estudiantes aplican y refuerzan los principales conceptos referidos al primer curso de programación. Se muestran los resultados obtenidos.*

Palabras clave: *Programación, Primer Curso, Juegos, Motivación.*

Abstract: *This paper describes our experience using Python as first programming language. We have designed a sequence of labs where using a game programming approach students apply and reinforce the main concepts regardless to the first course of programming. We show our results.*

KeyWords: *Programming, First Programming Course, Gaming, Motivation.*

1. Introducción

El cómo mantener interesados a los estudiantes en un curso de programación no es nuevo para los que enseñan la materia. Existen investigaciones que muestran que la utilización de gráficos y el desarrollo de juegos tienen gran impacto en el interés de los estudiantes [1,2,3,4,5,6,7].

Python [8] tiene muchas ventajas para enseñar el primer curso de programación. Su sintaxis sencilla y su modo interactivo hace que éste sea ideal para la enseñanza. El combinar Python con un enfoque orientado a juegos hace que los estudiantes se sientan motivados. La naturaleza visual de los juegos es conveniente para el aprendizaje, dado que es bastante fácil para los estudiantes el ver cómo un constructor es utilizado en un juego y ver los errores lógicos de su aplicación de forma visual [9].

En este artículo, presentamos nuestra experiencia en la utilización de Python [8] como primer leguaje de programación. Se resalta la secuencia de laboratorios orientados a la creación de juegos. Cabe resaltar que ya existen otras Universidades que migraron a Python para el primer curso de programación. Entre ellas se destaca la Universidad de Chile [11], Georgia Tech, USA [18] Chapman University, USA [19], USA, Saint Louis University, USA [20], Massachusetts Institute of Technology (MIT) [21], entre muchas otras.

El resto del artículo está organizado de la siguiente manera. En la sección 2, se presenta las generalidades del curso. En la sección 3, se presenta las características de Python. En la sección 4, se muestran la forma en que se diseñaron los laboratorios y los temas reforzados. En la sección 5, se presenta una discusión sobre los resultados obtenidos. Finalmente, en la sección 6, se presentan nuestras conclusiones y trabajo futuro.

2. El primer curso de programación

2.1 Generalidades

El primer curso de programación, denominado Introducción a la Programación, es dictado en el Primer Semestre de la Carrera de Ingeniería del Software de la Universidad La Salle de Arequipa [10]. El curso tiene una duración de 17 semanas. Tiene 8 horas semanales (2 horas teóricas, 2 horas prácticas y 4 horas de laboratorio). Se utiliza Python como lenguaje de programación desde el año 2012, pero recién desde el año 2013 se está utilizando el enfoque orientado a juegos.

2.2 Objetivos

El primer curso de programación tiene como objetivos que los estudiantes conozcan los constructores fundamentales de programación, apliquen los principios de abstracción para la creación de programas y desarrollen el pensamiento algorítmico

2.3 Contenidos

Los contenidos del curso se resumen en la Tabla 1.

Tabla 1. Contenidos del curso introducción a la Programación

1. Introducción
• Visión General: Algoritmos y programas Intérpretes y compiladores Paradigmas de programación. • Tipos de Datos: Enteros, Reales y Cadenas (String). • Sintaxis Básica de Programas Simples: Variables, Tipos, Expresiones y Asignaciones Entrada y Salida Simple, Variables. • Funciones predefinidas (impresión y captura de datos), caracteres especiales.

2. Condicionales e Iteración While
• Introducción a Módulos, funciones predefinidas,

sentencia import • Expresiones Condicionales • Álgebra Booleana y Operadores • Estructuras de Control Condicionales: if • Estructura de Control Condicionales if – else, if anidados • Estructura de Repetición: while y sentencia break
3. **Funciones**
• Definición y Uso de Funciones • Parámetros y Argumentos • La sentencia return • Llamados a funciones • Composición de Funciones • Alcances de Variables • Funciones Recursivas
4. Listas, Tuplas y Diccionarios
• Strings multilineas • Constantes • Definición de listas, acceso , modificación y eliminación de elementos • Lista de listas • Métodos predefinidos sobre listas • Funciones range() y list() • El iterador for • Equivalencia entre while y for • Aplicaciones con listas: Vectores. Métodos de búsqueda y de ordenación en vectores. • Matrices y operaciones sobre matrices. • Imágenes con Listas • Tuplas • Diccionarios: claves • Funciones y métodos predefinidos para diccionarios
5. **Proyecto de Aplicación**
• Manejo de Coordenadas • Librería Gráfica Pygame • Funciones de Dibujo • Colisión y Detección • Programación orientada eventos • Funciones de Sonido

3. Python

Python tiene muchas ventajas para la enseñanza del primer curso de programación [11,12,13]. En esta sección, resaltamos las principales características de Python.

3.1 Modo interactivo

Su sintaxis sencilla y su modo interactivo permiten al estudiante ver resultados de forma inmediata cuando se trata de probar nuevos conceptos. Así, por ejemplo, en la Figura 1 se muestra una prueba del funcionamiento de la función `random`.

```
>>> import random
>>> random.randint(1, 20)
12
>>> random.randint(1, 20)
18
>>> random.randint(1, 20)
3
>>> random.randint(1, 20)
18
>>> random.randint(1, 20)
7
>>>
```

Figura 1: Modo interactivo de Python.

3.2 Sintaxis sencilla

Python tiene una sintaxis regular simple. Las sentencias son terminadas por un salto de línea, y las estructuras de bloque son indicadas mediante identación. Estas características eliminan los clásicos errores de los programadores principiantes de olvidar puntos y comas y llaves de bloques. Por ejemplo, un error típico tanto en Java como en C++ es el no utilizar llaves para encerrar un bloque de más de dos instrucciones. En la Figura 2, se muestra este error en C++.

```
1  if (x < 0)
2      cout << "x es negativo";
3      x = -x;
```

Figura 2: Código C++: error en uso de llaves.

En Python, el código equivalente, mostrado en la Figura 3, se ejecuta sin problemas, dado que la misma identación determina el bloque.

```
1  if x < 0:
2      print "x es negativo"
3      x = -x
```

Figura 3: Código Python

Así mismo, al tener una sintaxis sencilla, Python permite centrarse más en la lógica y solución de problema que en detalles de sintaxis y conceptos avanzados antes de tiempo. Por ejemplo, considere el clásico programa que imprime "Hola Mundo". En la Figura 4, se muestra el código en Python.

```
1  print ('Hola Mundo')
```

Figura 4: Código Python "Hola Mundo".

C++ requiere que el programa sea parte de una función y sea precedido por una directiva preprocesada, como se ve en la Figura 5.

```
1  #include <iostream.h>
2  int main()
3  {
4      cout << "Hola Mundo!";
5  }
```

Figura 5: Código C++ "Hola Mundo".

En Java, la situación es aún peor. Todo el código debe estar dentro de una clase (tal como se muestra en la Figura 6). Además, se tiene el uso de los modificadores `public y static`, que generan dudas y preguntas en los estudiantes. Dudas difíciles de aclarar en los primeros días de clase.

```
1  public class HolaMundo
2  {
3     public static void main(String [] args)
4     {
5        System.out.println("Hola Mundo!");
6     }
7  }
```

Figura 6: Código Java "Hola Mundo".

3.3 No tipado

Python es un lenguaje no tipado. Esto es que no hay necesidad de declarar el tipo de dato de las variables. Esto reduce considerablemente la cantidad de código que el estudiante debe escribir y también elimina errores de tipo.

3.4 Constructores dundamentales

Algunas de las facilidades de Python que se presentan incluyen: variables, expresiones, operadores aritméticos, tipos, instrucciones de asignación y escritura (`print`). Funciones predefinidas (`input, abs, max, min, int, float`, funciones de las clases `math y random`) y funciones definidas por el programador. Instrucciones de selección (`if`) y repetición de instrucciones (`while`). Tipo bool: condiciones, operadores de comparación y operadores lógicos [14].

```
1  #Determina el Tipo de Rectangulo
2  #Encuentra la circunferencia circunscrita
3
4  print('Ingrese coordenada x1:')
5  x1 = input()
6  print('Ingrese coordenada y1:')
7  y1 = input()
8  print('Ingrese coordenada x2:')
9  x2 = input()
10 print('Ingrese coordenada y2:')
11 y2 = input()
12
13 altura = abs(int(y2) - int(y1))
14 base   = abs(int(x2) - int(x1))
15
16 if altura > base:
17     radio = base/2
18     print('Rectangulo es alto ')
19 if base > altura:
20     radio = altura/2
21     print('Rectangulo es bajo ')
22 if base == altura:
23     radio = altura/2
24     print('Rectangulo es cuadrado ')
25
26 area = 3.141516*(radio**2)
27 print('Area del circulo: ' + str(area))
```

Figura 7: Código Tipo de Rectángulo.

En la Figura 7, se muestran algunas de las características descritas previamente. El ejercicio pide al usuario ingresar dos coordenada del plano cartesiano (x, y) y determinar si es un tipo de rectángulo alto, bajo o cuadrado.

En la Figura 8, se muestra un programa que crea una función para calcular el máximo común divisor entre dos números mediante el algoritmo de Euclides. Este ejercicio resume las características básicas de la creación de funciones en Python.

```
1  def mcd(x,y):
2      while x!=y:
3          if x>y:
4              x=x-y
5          else:
6              y=y-x
7      return x
8
9  n = input("Ingrese n:")
10 m = input("Ingrese m:")
11 print (  mcd(n,m) )
```

Figura 8: Función Máximo Común Divisor.

Se aprecia la sencillez del encabezamiento de la función, que no necesita especificar el tipo del resultado ni de los parámetros. Al respecto, el tipo de una variable (`int` o `float`) se determina al momento de asignarle un valor entero o real. Por otra parte, los dos puntos causan que el editor de Python idente automáticamente las instrucciones de la función, mejorando notablemente su legibilidad.

Si bien, existen muchas más características de Python. El objetivo de este artículo no es dar un tutorial de Python, sinó mostrar el enfoque de programación orientado a la creación de juegos utilizando este lenguaje.

4. Diseño de Laboratorios

En esta sección se presenta el diseño de los laboratorios utilizados para el desarrollo del curso.

4.1 Fundamentos

En la primera semana se conoce el entorno básico de Pyton: variables, expresiones, asignaciones y operaciones aritméticas a través del Shell interactivo [14] (Figura 9).

```
>>> spam = 15
>>> spam + 5
20
>>> spam = 3
>>> spam + 5
8
>>>
```

Figura 9: Manejo de Variables y Operaciones Aritmeticas.

A continuación se inicia la creación de programas simples tal como se muestra en la Figura 10. Se resalta el uso de funciones predefinidas para la impresión en consola (`print`) y para la captura de datos (`input`).

```
# Este programa muestra
# la salida e ingreso de datos


print ('Hola')
print('Cual es tu Avatar?')
miAvatar = input()
print('Bienvenido ' + miAvatar + ' ,estas ')
print('listo para iniciar la aventura?')
```

Figura 10: Un primer programa.

4.2 Condicionales

En la Figura 11 se muestra el laboratorio que inicia el tema de condicionales. Inicialmente se presenta como debería funcionar el programa.

```
>>>
Hola!, cual es tu nombre ?
Elizabeth
 hola Elizabeth
estoy pensando en un numero entre 1 y 20...Adivina cual es!!!
6
lo siento Elizabeth tu suposicion es muy baja
el numero era el 7
>>>
```

Figura 11: Laboratorio Condicionales – Funcionamiento.

En la Figura 12, se presenta el código que implementa la funcionalidad mostrada en la Figura 11. Se resalta, además, del uso de condicionales, la aplicación de la función predefinida `random`, la cual es utilizada constantemente en la creación de juegos.

```
#Juego Adivina un numero
#JuevoAdivinaV0.py

import random

numeroSecreto = random.randint(1,20)

print('Hola!, cual es tu nombre? ')
nombre = input()
print('hola ' + nombre)
print('estoy pensando en un numero entre 1 y 20...Adivina cual es!!! ')
numeroStr = input()

numeroInt = int(numeroStr)
numeroSecretoStr = str(numeroSecreto)

if(numeroSecreto == numeroInt):
    print('felicitaciones ' + nombre + ' el numero secreto era: ' + numeroSecretoStr)

if(numeroSecreto < numeroInt):
    print('lo siento ' + nombre +  ' tu suposicion es muy alta')
    print('el numero secreto era: ' + numeroSecretoStr)

if(numeroSecreto > numeroInt):
    print('lo siento ' + nombre +  ' tu suposicion es muy baja')
    print('el numero secreto era: ' + numeroSecretoStr)
```

Figura 12. Laboratorios Condicionales - Código

4.3 Bucle While

Para el tema de bucles, se reutiliza el mismo ejercicio visto en Condicionales, con la diferencia que esta vez el usuario tiene hasta seis oportunidades para adivinar el número secreto. Además, el usuario recibe una ayuda en cuanto a si el número ingresado es muy alto o muy bajo. El funcionamiento se muestra en la Figura 13.

```
>>>
Hola!, cual es tu nombre?
Elizabeth
hola Elizabeth
estoy pensando en un numero entre 1 y 20.
Adivina cual es!!!
20
lo siento Elizabeth tu suposicion es muy alta
Adivina cual es!!!
2
lo siento Elizabeth tu suposicion es muy baja
Adivina cual es!!!
3
lo siento Elizabeth tu suposicion es muy baja
Adivina cual es!!!
19
lo siento Elizabeth tu suposicion es muy alta
Adivina cual es!!!
18
lo siento Elizabeth tu suposicion es muy alta
Adivina cual es!!!
16
lo siento Elizabeth tu suposicion es muy alta
No. El numero que estaba pensando era :  5
>>>
```

Figura 13: Laboratorio Bucle While – Funcionamiento.

El código que implementa el funcionamiento se muestra en la Figura 14.

```
#Juego Adivina un numero con varias opciones
#JuevoAdivinaV2.py

import random

numeroIntentos = 0

print('Hola!, cual es tu nombre? ')
nombre = input()

numeroSecreto = random.randint(1,20)

print('hola ' + nombre)
print('estoy pensando en un numero entre 1 y 20. ')

while numeroIntentos < 6:
    print('Adivina cual es!!!')
    numeroStr = input()
    numeroInt = int(numeroStr)

    numeroIntentos = numeroIntentos + 1

    if(numeroSecreto < numeroInt):
        print('lo siento ' + nombre +  ' tu suposicion
```

Figura 14: Laboratorio Bucle While – Código.

Es importante resaltar que durante este tema se realizan muchos ejercicios referidos a validación de ingresos de datos. Así, por ejemplo, en la Figura 15, se muestra el funcionamiento de la elección del símbolo para el juego de Tres en Raya [15]. Mientras el usuario no ingrese un símbolo válido, el juego sigue solicitando el ingreso. El código se puede apreciar en la Figura 16.

```
>>>
Bienvenido al Juego Tres en Raya!
Quiere ser X o O
A
Quiere ser X o O
B
Quiere ser X o O
U
Quiere ser X o O
X
jugador inicia el juego.
```

Figura 15: Laboratorio Bucle While – Validación de Ingreso.

4.4 Funciones

Para el tema de funciones, se reutilizan muchos de los ejercicios desarrollados previamente, pero esta vez se encapsulan en una función. Una muestra de ello se puede apreciar en las Figuras 16 y 17

```
def quienInicia():
    # Aleatoreamente decide
    #quien empezara el juego .

    if random.randint(0, 1) == 0:
        return 'computadora'
    else:
        return 'jugador'
```

Figura 16: Laboratorio Funciones – Código Función inicio del juego Tres en Raya.

```
23  def ingreseLetraJugador():
24      letra = ''
25      while not (letra == 'X' or letra == 'O'):
26          print('Quiere ser X o O ')
27          letra = input().upper()
28      if letra == 'X':
29          return ['X', 'O']
30      else:
31          return ['O', 'X']
```

Figura 17: Laboratorio Funciones – Código Función elección de Símbolo

4.5 Listas y Bucle For

Para el tema de listas, se utilizan dos juegos. El de Tres en Raya [15] y el Juego del Ahorcado [16]. En este artículo, mostramos, en la Figura 18, el funcionamiento del juego del Ahorcado. En las Figura 19, se muestra el código parcial que implementa la impresión de los espacios en blanco. En la Figura 20, se muestra el código parcial que pregunta si es que la letra ingresada es parte de la palabra secreta.

```
>>>
A H O R C A D O


  +---+
  |   |
      |
      |
      |
      |
==========

Letras Faltantes:

_ _ _ _ _ _ _ _ _ _ _ _ _ _
Ingrese una letra.
```

Figura 18: Laboratorio Listas – Funcionamiento Juego del Ahorcado.

```
83      #muestra la palabra secreta con
84      #espacios en blanco entre ellas
85      for letter in blanks:
86          print(letter, end=' ')
87      print()
```

Figura 19: Laboratorio Listas - Código Impresión Espacios en Blanco.

```
77      #remplaza espacios en blanco
78      #con las palabras correctamente adivinadas
79      for i in range(len(palabraSecreta)):
80          if palabraSecreta[i] in correctLetters:
81              blanks = blanks[:i] + palabraSecreta[i] +
82                                          blanks[i+1:]
```

Figura 20: Laboratorio Listas - Código Verificación si Letra Pertenece a Palabra Secreta.

4.6 Reforzando lo aprendido

Para reforzar los temas vistos previamente, desde el Semestre 2013 I, se ha hecho uso de la librería gráfica Pygame [9].

Para utilizar Pygame, solo se necesita conocer conceptos simples de manejo de coordenadas y pixeles, así como el conocimiento de algunas funciones básicas predefinidas de Pygame: creación de ventas, funciones de dibujo: círculos, rectángulos y líneas. Pygame contiene una serie de módulos que puede ser utilizado de forma independiente [17]. En la Figura 21, se muestra algunas de las formas que se crean cuando se inicia el trabajo con Pygame.

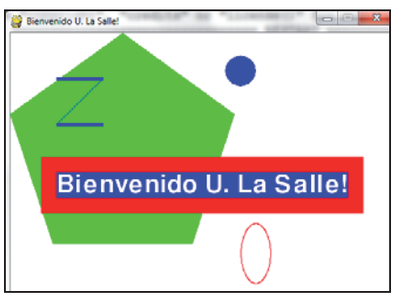

Figura 21: Laboratorio manejo de funciones de dibujo Pygame.

Para reforzar el tema de condicionales y bucles, se hace uso del movimiento de bloques en pantalla. Estos bloques representarían elementos de nuestro juego más adelante, tal como se muestra en la Figura 22.

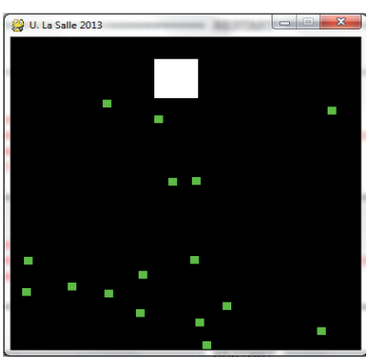

Figura 22: Movimiento de bloques.

Un bloque puede moverse en una de las cuatro direcciones diagonales: abajo-derecha, abajo-izquierda, arriba-derecha o arriba-izquierda. Cuando el bloque choca un borde de la ventana, éste debe rebotar en una nueva dirección diagonal. Existen ocho posibilidades en las que el bloque puede rebotar. Éstas se resumen en la Figura 23.

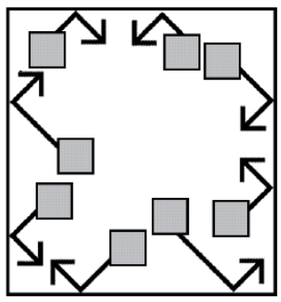

Figura 23: Diagrama de movimientos y rebotes diagonales.

Una parte del código que implementa este movimiento se muestra en la Figura 24. Se puede apreciar el uso intensivo del bloque `if`. De las líneas 78 a 91 se implementa el movimiento de los bloques en diagonal. De las líneas 92 a 112 se implementa el cambio de dirección del bloque cuando rebota en la ventana.

```
78    # mueve el jugador rebotando en las paredes
79    if jugador['dir'] == ABAJO_IZQ:
80        jugador['rect'].left =  jugador['rect'].left - VELOCIDAD_MOVI
81        jugador['rect'].top = jugador['rect'].top + VELOCIDAD_MOVI
82    if jugador['dir'] == ABAJO_DERECHA:
83        jugador['rect'].left = jugador['rect'].left +  VELOCIDAD_MOVI
84        jugador['rect'].top =  jugador['rect'].top +  VELOCIDAD_MOVI
85    if jugador['dir'] == ARRIBA_IZQUIERDA:
86        jugador['rect'].left = jugador['rect'].left -  VELOCIDAD_MOVI
87        jugador['rect'].top = jugador['rect'].top - VELOCIDAD_MOVI
88    if jugador['dir'] == ARRIBA_DER:
89        jugador['rect'].left = jugador['rect'].left +  VELOCIDAD_MOVI
90        jugador['rect'].top = jugador['rect'].top -  VELOCIDAD_MOVI
91
92    # verifica si el jugador se ha movido fuera de la ventana
93    if jugador['rect'].top < 0:
94        # el jugador se ha movido por encima
95        if jugador['dir'] == ARRIBA_IZQUIERDA:
96            jugador['dir'] = ABAJO_IZQ
97        if jugador['dir'] == ARRIBA_DER:
98            jugador['dir'] = ABAJO_DERECHA
99
100   if jugador['rect'].bottom > ALTO_VENTANA:
101       # jugador se ha movido por debajo de la ventana
102       if jugador['dir'] == ABAJO_IZQ:
103           jugador['dir'] = ARRIBA_IZQUIERDA
104       if jugador['dir'] == ABAJO_DERECHA:
105           jugador['dir'] = ARRIBA_DER
106
107   if jugador['rect'].left < 0:
108       # el jugador se ha movido pasando el lado izquierdo
109       if jugador['dir'] == ABAJO_IZQ:
110           jugador['dir'] = ABAJO_DERECHA
111       if jugador['dir'] == ARRIBA_IZQUIERDA:
112           jugador['dir'] = ARRIBA_DER
```

Figura 24: Código Parcial del Movimiento de Bloques y sus Rebotes.

Un comportamiento bastante común en los juegos es la detección de colisiones. Detectar una colisión puede provocar incrementar puntaje, perder puntaje, ganar puntos adicionales o perder una vida. Así mismo, una colisión hace que el objeto objetivo desaparezca. La implementación de la detección de colisiones nos permite reforzar el tema de funciones. Esto se puede apreciar en la Figura 25.

```
5   def rectangulosSeSobreponen(rect1, rect2):
6       for a, b in [(rect1, rect2), (rect2, rect1)]:
7           # Verifica si las esquinas de a  estan dentro de las esquinas de b
8
9           if ((puntoDentroDeRectangaulo(a.left, a.top, b)) or
10              (puntoDentroDeRectangaulo(a.left, a.bottom, b)) or
11              (puntoDentroDeRectangaulo(a.right, a.top, b)) or
12              (puntoDentroDeRectangaulo(a.right, a.bottom, b))):
13              return True
14
15      return False
16
17  def puntoDentroDeRectangaulo(x, y, rect):
18      if (x > rect.left) and (x < rect.right) and (y > rect.top) and (y < rect.bottom):
19          return True
20      else:
21          return False
```

Figura 25: Aplicación de funciones: código implementación detección de colisiones.

Una vez dominados los principios de dibujo y movimiento. Se les enseñó a los estudiantes cómo "disfrazar" los bloques con imágenes. Así mismo se les enseñó cómo utilizar sonido y teclado (una visión muy genérica de programación orientada a eventos). Algunas de las aplicaciones realizadas por los estudiantes van desde una simulación del juego Pacman (Figura. 26), hasta la implementación de un juego con puntaje y vidas (Figura 27).

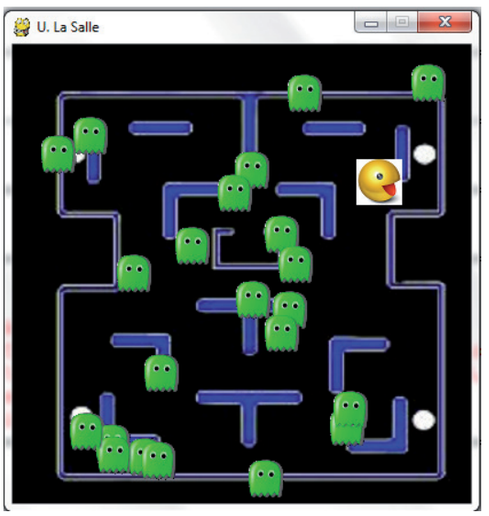

Figura 26: Ejemplo de Aplicación 1: Simluación Inicial de Pacman.

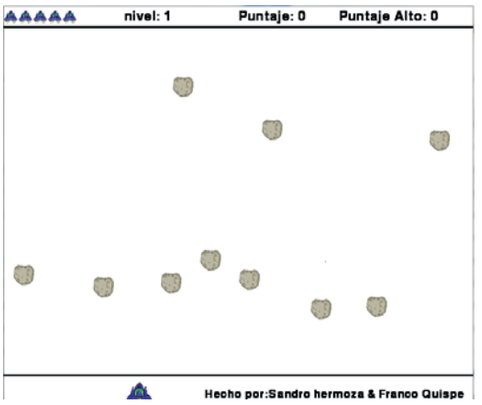

Figura 27: Ejemplo de Aplicación 2: Juego Invasores Completo.

5. Análisis de resultados

Desde el punto de vista del docente, el utilizar Python permitió al docente centrarse más en ayudar a los estudiantes a comprender los constructores fundamentales y desarrollar su pensamiento algorítmico. Las características técnicas de Python facilitan el aprendizaje de los estudiantes haciendo la programación más sencilla y entretenida permitiendo incursionar en conceptos computacionales más avanzados (uso de librería gráfica). El enfoque orientado a la creación de juegos y uso de Pygame logró motivar a los estudiantes y reforzar los conceptos aprendidos durante las primeras 11 semanas.

El punto de vista de los estudiantes se obtuvo a través de una encuesta aplicada al final del semestre. La encuesta se aplicó a 1os 19 estudiantes del curso. Las alternativas de respuesta fueron: (A) Totalmente de Acuerdo, (B) La mayor parte del tiempo (C) Algunas veces, (D) Casi nada, (E) En desacuerdo. El resultado de las encuestas se resume en la Tabla 2.

Tabla II, Resultado de las Encuestas.

1.	¿Considera que el programar juegos lo ayudó a comprender mejor los constructores fundamentales de programación (if, while, for,)? A (68%) B (26%) C (5%) D (0%) E (0%)
2.	¿Considera que programando juegos se comprende mejor el concepto de funciones? A (53%) B (32%) C (16%) D (0%) E (0%)
3.	¿Considera que las asignaciones desarrolladas en Modo Gráfico son más divertidas que las asignaciones desarrolladas en modo consola? A (53%) B (37%) C (5%) D (0%) E (5%)
4.	¿Considera que usted tuvo más dedicación y esfuerzo en las asignaciones realizadas en Modo Gráfico que con las asignaciones realizadas en modo consola? A (35%) B (29%) C (29%) D (0%) E (6%)
5.	¿Considera que comprender Pygame no fue complicado? A (16%) B (37%) C (32%) D (11%) E (5%)
6.	¿Considera que utilizar Pygame incentivó su interés por aprender nuevas cosas por iniciativa propia? A (58%) B (26%) C (16%) D (0%) E (0%)

De acuerdo con el resultado obtenido en las encuestas, el enfoque orientado a juegos y el uso Pygame también resultó positivo para los estudiantes. De la pregunta 5, se puede apreciar que para los estudiantes aprender Pygame no es tan simple. Esto requiere un mayor esfuerzo del docente para poder presentarlo de manera más sencilla en el siguiente semestre.

6. Conclusiones

El primer curso de programación de nuestra carrera tiene como objetivo que los estudiantes conozcan los constructores fundamentales de programación, apliquen los principios de la matemática para crear programas, apliquen los principios de abstracción para la creación de programas y desarrollen el pensamiento algorítmico. Como educadores buscamos que nuestros estudiantes se sientan motivados y logren dichos objetivos. Existen dos decisiones importantes, el lenguaje a utilizar y el enfoque de enseñanza. En nuestra Universidad, se decidió utilizar Python como lenguaje de programación y un enfoque orientado a la creación de juegos soportado con la librería gráfica Pygame.

Hemos desarrollado una serie de laboratorios que han permitido enseñar los constructores fundamentales a través del contexto del desarrollo de juegos. La naturaleza visual de los juegos fue conveniente para el aprendizaje, dado que fue sencillo para los estudiantes el ver cómo un constructor es utilizado en un juego y ver los errores lógicos de su aplicación de forma visual.

Las encuestas han mostrado la aceptación del enfoque orientado a juegos y del uso de Pygame por parte de nuestros estudiantes. Así mismo, la calidad de los juegos presentados como trabajo de fin de curso confirma la motivación y el interés generado con el uso de la librería gráfica Pygame.

Como trabajo futuro, se pretende experimentar el uso de Python para la enseñanza de la Programación Orientada a Objetos. Así mismo, se busca profundizar en el uso de Pygame como elemento motivador.

Referencias bibliográficas

[1] E. Sweedyk, M. deLaet, M. C. Slattery, and J. Kuffner, "Computer games and CS education: why and how," in Proceedings of the 36th SIGCSE technical symposium on Computer science education , St. Louis, Missouri, USA , 2005, pp. 256-257.

[2] U. Wolz, T. Barnes, I. Parberry, and M. Wick, "Digital gaming as a vehicle for learning," ACM SIGCSE Bulletin, vol. 38, no. 1, pp. 394-395, 2006.

[3] J. D. Bayliss and S. Strout, "Games as a "flavor" of CS1," ACM SIGCSE Bulletin, vol. 38, no. 1, pp. 500-504, Mar. 2006.

[4] S. Leutenegger and J. Edgington, "A games first approach to teaching introductory programming," ACM SIGCSE Bulletin, vol. 39, no. 1, pp. 115-118, Mar. 2007.

[5] J.C. Adams, Chance-it: an OO capstone project for cs-1, SIGCSE'98, 10-14, 1998.

[6] K.. Becker, Teaching with games: the minesweeper and asteroids experience, J. Comput. Small Coll, 17(2), 23-33, 2001

[7] T. Lorenzen, W. Heilman, Cs1 and cs2: write computer games in java! SIGCSE Bull., 34(4), 99-100, 2002.

[8] Python. http:// www.python.org

[9] Pygame. http:// www.pygame.org

[10] Universidad La Salle, Arequipa. http:// www.ulasalle.edu.pe

[11] J. Alvarez. Python en un primer curso de

computación XI V Congreso Chileno de Educación Superior en Computación"; Valparaiso; noviembre 2012

[12] J. M. Zelle. *Python as a First Language*, 13th Annual Midwest Computer Conference, March 1999.

[13] H. Wang, "From C to Python: Impact on Retention and Student Performance," in Proceedings of The 2009 International Conference on Frontiers in Education: Computer Science and Computer Engineering (FECS), Las vegas, USA, 2009, pp. 170-174.

[14] D. Allen. "Think Python – How to think like a Computer Scientist". Green Tea Press. 2008

[15] Tres en Linea.
http://es.wikipedia.org/wiki/Tres_en_l%C3%ADnea

[16] Juego Ahorcado.
http://es.wikipedia.org/wiki/Ahorcado_%28juego%29

[17] W. McGugan, Beginning Game Development with Python and Pygame: From Novice to Professional. Apress, 2009.

[18] D. Ranum, B. Miller, J. Zelle, and M. Guzdial. Successful Approaches to Teaching Introductory Computer Science Courses with Python. In SIGCSE '06: Proceedings of the 37th SIGCSE Technical Symposium on Computer Science Education (March 2006), pp. 396–397

[19] A. Radenski. "Python first": A lab-based digital introduction to computer science. In Proc. 11th Annual Conf. on Innovation and Technology in Computer Science (ITiCSE), pages 197–201, Bologna, Italy, June 2006.

[20] M. H. Goldwasser and D. Letscher. Teaching an Object-Oriented CS1 with Python**,** *Proceedings of the 13th Annual Conference on Innovation and Technology in Computer Science (ITiCSE)*, Madrid, Spain Jun. 30-Jul. 2, 2008, pp. 42-46.

[21] Daher, W. S. *EECS Revamps Course Structure*.
http://tech.mit.edu/V125/N65/coursevi.html
Accedido el 27 de agosto del 2013.