UNIVERSIDAD MAYOR DE SAN ANDRÉS

FACULTAD DE CIENCIAS PURAS Y NATURALES

CARRERA DE INFORMÁTICA

PERFIL DE TESIS DE GRADO

“HERRAMIENTA CASE PARA GENERAR CÓDIGO PYTHON A PARTIR DE UN PSEUDCÓDIGO”

POSTULANTE: TERRAZAS PAZ IVÁN ARAMÍS

TUTOR METODOLÓGICO: M.SC. GROVER RODRIGUEZ RAMIREZ

ASESOR: PH.D. YOHONI CUENCA SARZURI

LA PAZ – BOLIVIA
2020

Índice de Perfil

1. Introducción 1

2. Problema 2

2.1. Antecedentes del Problema 2

2.2. Formulación del Problema 3

3. Objetivos 3

3.1. Objetivo General 3

3.2. Objetivos Específicos 3

4. Hipótesis 4

5. Justificaciones 4

5.1. Justificación Social 4

5.2. Justificación Económica 4

5.3. Justificación Tecnológica 4

5.4. Justificación Científica 5

6. Alcances y Limites 5

6.1. Alcances 5

6.2. Limites 5

7. Marco Teórico 6

7.1. Herramienta CASE 6

7.1.1. Características de la Herramienta CASE 6

7.1.2. Uso de Herramientas CASE 8

7.1.3. Clasificación de las Herramientas CASE 10

7.1.4. Ventajas y Desventajas de usar Herramientas CASE 11

7.1.4.1. Ventajas 11

7.1.4.2. Desventajas 11

7.2. Pseudocódigo 12

7.3. Python 14
7.3.1. Elementos de Python 15
7.3.1.1. Variables 15
7.3.1.2. Tipos de Datos 16
7.3.1.3. Operadores Aritméticos 18
7.3.1.4. Comentarios 19

8. Cronograma 20

9. Índice Tentativo 21

10. Bibliografía 22

1. Introducción

El mundo va avanzando a pasos agigantados en la tecnología y cada vez se necesitan más informáticos que puedan dominar varios lenguajes de programación y poder aplicar en el marco laboral. Pero, así como crece el mundo tecnológico, también nacen nuevos lenguajes y métodos para poder desarrollar programas. Por lo tanto, un informático debe elaborar software de calidad a un buen precio y en un corto tiempo, de acuerdo a las exigencias del cliente, de tal manera que lo desarrollado sea compatible para la mayoría de las plataformas.

Para poder lograr este tipo de exigencias, se hace uso de una herramienta CASE, la cual nos ayudara con su ambiente grafico de diseño, automatizando y ordenando nuestro código para el cumplimiento de los estándares recomendados por la ingeniería de software.

Un buen desarrollador de sistemas debe ser polifuncional, y no centrarse en un solo lenguaje de programación, debe estar listo para desarrollar soluciones capaces de ejecutarlas en diferentes arquitecturas, sin que esto requiera cambio de la estructura del sistema maestro.

Con el objetivo de solucionar estos problemas podemos recurrir a un solo software que nos ayude a desarrollar aplicaciones, y sea flexible a la hora de escoger el lenguaje de programación; todo esto a partir de conocimiento básico como el pseudocódigo.

2. Problema

2.1. Antecedentes del Problema

Para esta tesis se hizo una búsqueda y análisis de trabajos similares que trata de cumplir un objetivo parecido al que se plantea en este trabajo:

"Herramienta CASE para graficar y documentar diagramas UML 2.2", Uriol y Cruz (2012) – Facultad de Ciencias Físicas y Matemáticas - Universidad Nacional de Trujillo, se destaca que se ha desarrollado la herramienta CASE para graficar y documentar diagramas UML 2.2, basad en el software libre y donde se ha puesto especial cuidado en el diseño de una interfaz de usuario lo más fluida, intuitiva y atractiva posible.

"Generaciones de Soluciones Multicapa y Multiplataforma por medio de Herramientas CASE", Guerra (2011) – Facultad de Ingeniería en Ciencias Aplicadas – Universidad Técnica del Norte, las herramientas CASE maximizan la calidad y la productividad del software generado y de todo el equipo de desarrollo. Ya que no está frente a una herramienta común de programación sino frente a una metodología de diseño de aplicaciones; que está enfocada a crear una cultura de programación ordenada y sistemática tanto en los sistemas a desarrollar como en el personal desarrollo.

"Diseño e implementación de una herramienta CASE para la metodología programación extrema", Polo (2016) – Facultad de Ingeniería eléctrica, electrónica, informática y mecánica - Universidad Nacional de San Antonio Abad del Cusco, el presente trabajo de tesis realiza el diseño e implementación de una herramienta CASE para apoyar en las labores de planificación, monitoreo y documentación de un determinado equipo de trabajo que utilice la metodología ágil denominada programación extrema durante la elaboración de un determinado proyecto de desarrollo de software.

El inicio del mundo de la informática, más específicamente de un desarrollador, es aprender a tener lógica para resolver problemas mediante estructuras y diagramas, que son comunes para todos los lenguajes de programación. Pero el problema sería que cada lenguaje de programación tiene su sintaxis para poder usarlas. Existen muchos lenguajes de

programación de los cuales no hay favoritos, ya que un año puede estar entre los más usados algún lenguaje y al otro año tomaría su lugar otro lenguaje de programación y tendría más ventajas sobre el primero, entonces tener q migrar nuestro software puede resultar muy complicado o lo cual nos llevaría a desarrollar de nuevo desde un inicio con la nueva sintaxis.

Aprender un lenguaje de programación no suele ser tan complicado, pero si uno quiere especializarse y dominar un lenguaje entonces toma su tiempo. Por lo cual es complicado estar aprendiendo varios lenguajes. Como solución habría dos: una seria contratar varios desarrolladores que dominen distintos idiomas, y otra opción sería desarrollar a base de una Herramienta CASE que sea capaz de generar todas las aplicaciones para satisfacer los requisitos.

El pseudocódigo es un lenguaje que describirá nuestro algoritmo, para poder desarrollar nuestra solución en papel, teniendo los conocimientos básicos de programación. Este pseudocódigo no lo podrá leer la máquina, solo el ser humano, es como una manera común para comunicarse unos con otros.

2.2. Formulación del Problema

¿De qué manera se puede generar código Python a partir de un pseudocódigo universal?

3. Objetivos

3.1. Objetivo General

Desarrollar una herramienta CASE para generar código Python a partir de un solo pseudocódigo que sea compatible con todos los lenguajes de programación.

3.2. Objetivos Específicos

- Analizar y desarrollar la herramienta CASE.
- Verificar la Sintaxis de Python.
- Analizar el uso de un pseudocódigo universal, para no tener complicaciones con desarrollar en la Herramienta CASE.
- Evaluar el uso de esta herramienta.

- Implementar esta herramienta usando como lenguaje de programación a Python.

4. Hipótesis

A partir del desarrollo de una Herramienta CASE se puede generar código Python mediante un pseudocódigo universal.

5. Justificaciones

5.1. Justificación Social

Socialmente ayudara a todas las personas a que puedan desarrollar sus aplicaciones sin necesidad de tener que dominar todos los lenguajes, simplemente aprender las estructuras y la lógica de estas; y lo más importante saber el pseudocódigo que usa la herramienta y así de manera independiente poder desarrollar nuestra aplicación.

5.2. Justificación Económica

Con el fin de eliminar el lucro y hacer un aporte que realmente beneficia a los estudiantes universitarios que actualmente cursan las carreras de programación se desarrolló este software con el objetivo de contribuir y alcanzar las metas tratando de no invertir un gasto económico considerable en la investigación, desarrollo e implementación para tal cometido se buscara para la recolección de datos a personas que no ostenten intereses personales o institucionales ya que terminado el software este será gratuito y estará a libre disponibilidad de todas las personas que deseen usarlas porque será fruto de las herramientas, métodos y pruebas accesibles de dominio y conocimiento público.

5.3. Justificación Tecnológica

Se usará como referencia a otras Herramientas CASE que cumplan un rol similar al de este trabajo, ya que no es necesario reinventar la herramienta mencionada, sino que sirve de inspiración y modelo para desarrollar una herramienta CASE. Por lo tanto se cuenta con herramientas similares para su referencia.

5.4. Justificación Científica

La implementación de una Herramienta CASE para generar lenguajes de programación, inicialmente Python propuesto en este trabajo, despierta el interés de la población en general sobre desarrollar software de manera más rápida, por lo que este trabajo pretende ser base para que en futuras investigaciones se puedan generar otros lenguajes de programación a base del pseudocódigo que maneje esta Herramienta CASE.

6. Alcances y Limites

6.1. Alcances

Este trabajo se centrará principalmente en ayudar a las personas que están empezando en el mundo del desarrollo y creación de software.

Entre las principales tareas a realizar, tendremos:

- Creación de funciones y procedimientos.
- Soporte de todas las estructuras algorítmicas teniendo en cuenta su inicio y final.
- Traducción de pseudocódigo al lenguaje de programación Python.
- Soporte de todos los tipos de variables.

6.2. Limites

- El software planteado no va verificar ni calificar la lógica del programador.
- No resolverá, ni sugerirá automáticamente los problemas algorítmicos.

7. Marco Teórico

A continuación, se presentará conceptos que son importantes en el desenvolvimiento de esta Tesis.

7.1. Herramienta CASE

Sommerville (2005) indica que la Herramienta CASE[1] comprende un amplio abanico de diferentes tipos de programas que se utilizan para ayudar a las actividades del proceso del software, como el análisis de requerimientos, el modelado de sistemas, la depuración y las pruebas. En la actualidad, todos los métodos vienen con tecnología CASE asociada, como los editores para las notaciones utilizadas en el método, módulos de análisis que verifican el modelo del sistema según las reglas del método y generadores de informes que ayudan a crear la documentación del sistema. Las herramientas CASE también incluyen un generador de código que automáticamente genera código fuente a partir del modelo del sistema y de algunas guías de procesos para los ingenieros de software.

7.1.1. Características de la Herramienta CASE

James (2001) plantea las características comunes de una Herramienta CASE, entre las cuales tenemos:

- Operaciones Iniciales

Los sistemas CASE almacenan información por proyecto, cada aplicación es considerada como un proyecto. La información que describe cada proyecto se mantiene por separado de la de otros.

- Menú Principal de Funciones

Muchas herramientas CASE permiten que el usuario seleccione una acción señalando su nombre o un numero sobre la pantalla, ya sea a través de un dispositivo apuntador o por el

[1] Computer Aided Software Engineering o Ingeniería de Software asistida por computadora

posicionamiento de una barra luminosa por medio de las teclas de flechas y tabulador contenidas en el teclado.

- Dibujo de Diagramas de flujo de datos

Los diagramas de flujo de datos son uno de los muchos tipos de diagramas y cartas disponibles en las herramientas CASE, es muy sencillo modificar los diagramas.

- Diccionario por proyecto

A medida que se formulan las especificaciones y la documentación, toda la información con respecto al proyecto se acumula en el diccionario de datos. Parte de la información, la graba directamente la persona que hace uso de la herramienta, otra parte es grabada automáticamente.

Una vez que la información se encuentra en el diccionario, puede volver a ser utilizada por el mismo proyecto en forma repetida, sin necesidad de definirla de nuevo.

Dentro del diccionario, las entradas se pueden añadir, modificar, listar, borrar y cambiar nombre. También es posible enlistar el contenido del diccionario con informes preformateados. Se tiene acceso a la información contenida en el diccionario desde cualquier parte.

- Pantallas e informes

Varias herramientas CASE, proporcionan un método rápido y sencillo para desarrollar prototipos de pantallas para que los usuarios finales trabajen con ellas. El analista puede diseñar y ejecutar pantallas y reportes con el apoyo de un menú, definir la distribución de una pantalla o reporte, el analista puede generar un reporte basándose en datos de prueba proporcionados al sistema.

- Herramientas para análisis y documentación

Algunas herramientas CASE ofrecen características tales como un conjunto de reportes que validan las descripciones del sistema.

- Utilerías

La información utilizada por el sistema se encuentra descrita por las funciones de utilería, estas funciones permiten definir las contraseñas de los usuarios, los privilegios de acceso y los procedimientos de respaldo. Las utilerías también proporcionan funciones de respaldo y recuperación. Con ellas es posible copiar o volver a crear una parte o todo el diccionario del proyecto de una copia de respaldo.

7.1.2. Uso de Herramientas CASE

De acuerdo a Kendall & Kendall (2011) las herramientas CASE se crearon de manera explícita para mejorar el trabajo rutinario a través del uso del soporte automatizado. Los analistas emplean herramientas CASE para aumentar la productividad, comunicarse con los usuarios de una manera más efectiva e integrar el trabajo que realizan en el sistema, desde el inicio hasta el fin del ciclo de vida.

Visible Analyst (VA) es un ejemplo de herramienta CASE que permite a los analistas de sistemas realizar planificación, análisis y diseño en forma gráfica para crear bases de datos y aplicaciones cliente/servidor complejas. Visible Analyst, aunado a otro producto de software conocido como Microsoft Visio, permite a los usuarios dibujar y modificar diagramas con facilidad.

Los analistas y usuarios en general reportan que las herramientas CASE les ofrecen un medio de comunicación relacionado con el sistema durante su conceptualización. Mediante el uso de soporte automatizado que incluye resultados en pantalla, los clientes pueden ver de inmediato la forma en que fluyen los datos y cómo se representan otros conceptos del sistema, para así poder solicitar correcciones o modificaciones que hubieran requerido de mucho más tiempo si se utilizaran herramientas anteriores.

Algunos analistas marcan la diferencia entre las herramientas CASE superiores e inferiores. Una herramienta CASE superior permite al analista crear y modificar el diseño del sistema. Toda la información sobre el proyecto se almacena en una enciclopedia conocida como repositorio CASE, una extensa colección de registros, elementos, diagramas, pantallas,

informes y demás información relacionada como se muestra en la Figura 1. Es posible producir informes del análisis mediante el uso de la información del repositorio para mostrar en qué partes está incompleto el diseño o dónde hay errores. Las herramientas CASE superiores también ayudan a sustentar el modelado de los requerimientos funcionales de una organización, auxiliar a los analistas y usuarios para dibujar los límites de un proyecto dado y ayudarlos a visualizar la forma en que el proyecto encaja con otras partes de la organización.

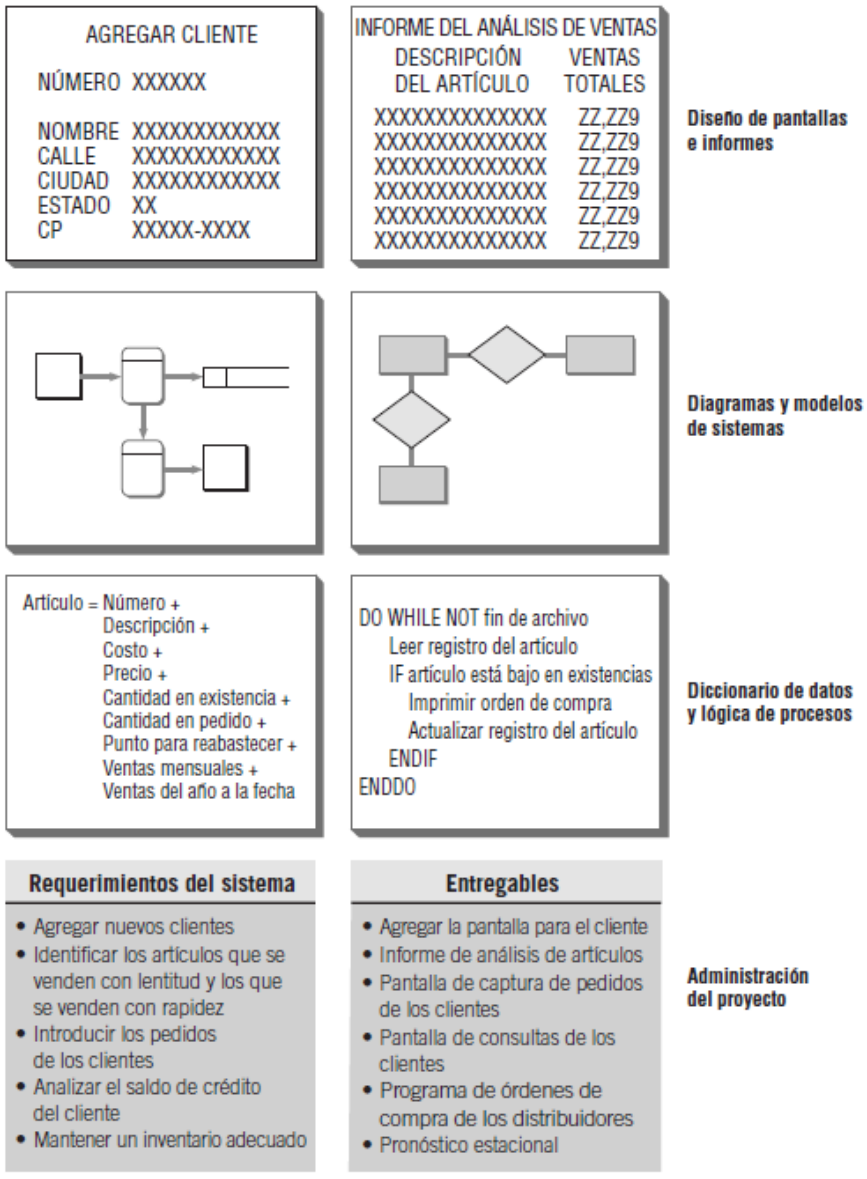

Figura 1. El concepto de Repositorio
Fuente: Kendall & Kendall (2011)

Las herramientas CASE inferiores se utilizan para generar código fuente de computadora, con lo cual se elimina la necesidad de programar el sistema. La generación de código ofrece varias ventajas:

1) El sistema se puede producir con más rapidez que si se escribieran programas computacionales.
2) La cantidad de tiempo invertido en el mantenimiento se reduce con la generación de código.
3) Se puede generar código en más de un lenguaje computacional, por lo que es más sencillo migrar los sistemas de una plataforma a otra.
4) La generación de código provee una manera efectiva en costo de personalizar los sistemas que se compran a terceros distribuidores para ajustarlos a las necesidades de la organización.
5) El código generado está libre de los errores típicos de los programas computacionales.

7.1.3. Clasificación de las Herramientas CASE

El Instituto Nacional de Estadística e Informática (1999) indica que no existe una única clasificación de herramientas CASE y, en ocasiones, es difícil incluirlas en una clase determinada. Podrían clasificarse atendiendo a:

- Las plataformas que soportan.
- Las fases del ciclo de vida del desarrollo de sistemas que cubren.
- La arquitectura de las aplicaciones que producen.
- Su funcionalidad.

Las herramientas CASE, en función de las fases del ciclo de vida abarcadas, se pueden agrupar de la forma siguiente:

a. **Herramientas integradas, I-CASE[2]**: abarcan todas las fases del ciclo de vida del desarrollo de sistemas. Son llamadas también CASE workbench.
b. **Herramientas de alto nivel, U-CASE[3]** o front-end, orientadas a la automatización y soporte de las actividades desarrolladas durante las primeras fases del desarrollo: análisis y diseño.
c. **Herramientas de bajo nivel, L-CASE[4]** o Back-end, dirigidas a las últimas fases del desarrollo: construcción e implantación.
d. **Juegos de herramientas o Tools-Case**, son el tipo más simple de herramientas CASE. Automatizan una fase dentro del ciclo de vida. Dentro de este grupo se encontrarían las herramientas de reingeniería, orientadas a la fase de mantenimiento.

7.1.4. Ventajas y Desventajas de usar Herramientas CASE

Según Garmire (1999) encuentra las siguientes ventajas y desventajas al momento de hacer uso de las Herramientas CASE.

7.1.4.1. Ventajas

- Facilidad para la revisión de aplicaciones
- Ambiente interactivo en el proceso de desarrollo
- Reducción de costos en desarrollo
- Incremento de la productividad

7.1.4.2. Desventajas

- El costo de adquisición
- Preparación del personal.
- Métodos estructurados.
- Falta de estandarización para el intercambio de información.
- Función limitada

[2] Integrated CASE, CASE integrado
[3] Upper CASE - CASE superior
[4] Lower CASE - CASE inferior

7.2. Pseudocódigo

En el ámbito de la programación, el Pseudocódigo es una de las técnicas más utilizadas para el diseño de Algoritmos. Pinales & Velázquez (2014) expresan que esta herramienta permite pasar casi de manera directa la solución del problema a un lenguaje de programación específico. El pseudocódigo es una serie de pasos bien detallados y claros que conducen a la resolución de un problema.

Flores (2005) cita en su libro las siguientes pautas básicas:

- Todo algoritmo debe tener un nombre, el cual deberá comenzar con una letra mayúscula. Si es un nombre ocmpuesto, la primera letra de cada palabra simple deberá estar en mayúsculas. No se permiten los espacios en blanco en el nombre del algoritmo.
- Es necesario que se determinen los datos de entrada y la información de la salida.
- Para declarar una variable ”x” se deberá determinar qué tipo de dato se almacenará.
- Para asignar un valor a una variable “x” se utiliza el signo igual.
- Para iniciar que la computadora lea un valor desde eun dispositivo externo y lo almacene en la variable “z”, se utiliza: LEER z.
- Para indicar que la computadora escriba hacia un dispositivo externo:
    - Para escribir un mensaje se utiliza: ESCRIBIR “Hola”
    - Para escribir el valor de una variable se utiliza: ESCRIBIR x.
    - Para escribir el resultado de una expresión se utiliza: ESCRIBIR x+2.

En la Figura 2 se puede observar la estructura básica de un pseudocódigo.

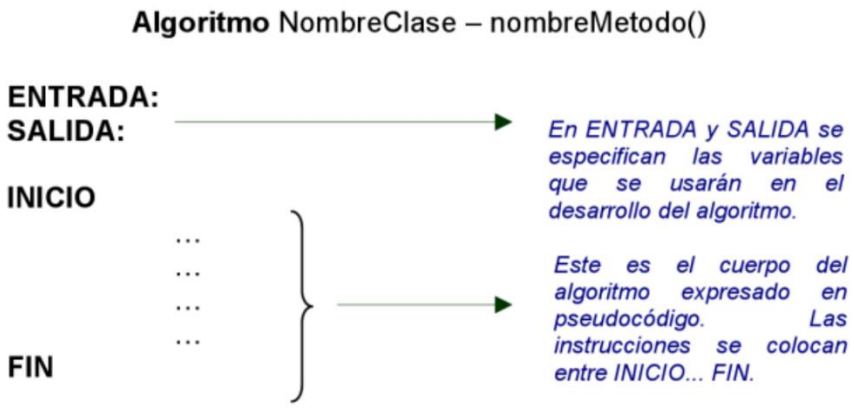

Figura 2: Estructura Básica de un Pseudocódigo
Fuente: Flores (2005)

Según Joyanes (2008) el pseudocódigo nació como un lenguaje similar al inglés y era un medio de representar básicamente las estructuras de control de programación estructurada que se verán en capítulos posteriores. Se considera un primer borrador, dado que el pseudocódigo tiene que traducirse posteriormente a un lenguaje de programación. El pseudocódigo no puede ser ejecutado por una computadora. La ventaja del pseudocódigo es que en su uso, en la planificación de un programa, el programador se puede concentrar en la lógica y en las estructuras de control y no preocuparse de las reglas de un lenguaje específico. Es también fácil modificar el pseudocódigo si se descubren errores o anomalías en la lógica del programa, mientras que en muchas ocasiones suele ser difícil el cambio en la lógica, una vez que está codificado en un lenguaje de programación. Otra ventaja del pseudocódigo es que puede ser traducido fácilmente a lenguajes estructurados como Pascal, C, C++, Java, C#, etc. El pseudocódigo original utiliza para representar las acciones sucesivas palabras reservadas en inglés —similares a sus homónimas en los lenguajes de programación—, tales como start, end, stop, if-then-else, while-end, repeat-until, etc. La escritura de pseudocódigo exige normalmente la indentación[5] de diferentes líneas. Una representación en pseudocódigo —en inglés— de un problema de cálculo del salario neto de un trabajador como muestra la Figura 3.

[5] Sangría en el margen izquierdo

```
start
  //cálculo de impuesto y salarios
  read nombre, horas, precio
  salario ← horas * precio
  tasas ← 0,25 * salario
  salario_neto ← salario - tasas
  write nombre, salario, tasas, salario
end
```

Figura 3: Ejemplo de Pseudocódigo
Joyanes (2008)

El algoritmo comienza con la palabra start[6] y finaliza con la palabra end[7], en inglés. Entre estas palabras, sólo se escribe una instrucción o acción por línea. La línea precedida por // se denomina comentario. Es una información al lector del programa y no realiza ninguna instrucción ejecutable, sólo tiene efecto de documentación interna del programa. Algunos autores suelen utilizar corchetes o llaves. No es recomendable el uso de apóstrofos o simples comillas como representan en algunos lenguajes primitivos los comentarios, ya que este carácter es representativo de apertura o cierre de cadenas de caracteres en lenguajes como Pascal o FORTRAN, y daría lugar a confusión.

7.3. Python

Fernandez (2012) describe a Python como un lenguaje de programación de alto nivel[8], interpretado y multipropósito. En los últimos años su actualización ha ido constantemente creciendo y en la actualidad es uno de los lenguajes de programación más empleados para el desarrollo de software.

Python puede ser utilizado en diversas plataformas y sistemas operativos, entre los que podemos destacar los más populares, como Windows, Mac OS X y Linux. Pero, además, Python puede funcionar en smartphones, Nokia desarrolló un intérprete de este lenguaje para su sistema operativo Symbian.

[6] En español, inicio
[7] En español, inicio
[8] Tipo de lenguaje de programación que permite al programador escribir programas y/o algoritmos que son independientes de un tipo particular de computadora.

Figura 4: Logo de Python
Fuente: Página oficial de Python[9]

Entre las principales razones para elegir Python, son muchos los que argumentan que sus principales características lo convierten en un lenguaje muy productivo. Se trata de un lenguaje potente, flexible y con una sintaxis clara y concisa. Además, no requiere dedicar tiempo a su compilación debido a que es interpretado.

Python es open source[10], cualquiera puede contribuir a su desarrollo y divulgación. Además, no es necesario pagar ninguna licencia para distribuir software desarrollado con este lenguaje. Hasta su intérprete se distribuye de forma gratuita para diferentes plataformas.

7.3.1. Elementos de Python

Como en la mayoría de los lenguajes de programación de alto nivel, en Python se compone de una serie de elementos que alimentan su estructura. Entre ellos, Bahit (2012) hace mención de los siguientes:

7.3.1.1. Variables

Una variable es un espacio para almacenar datos modificables, en la memoria de un ordenador. En Python, una variable se define con la sintaxis:

nombre_de_la_variable = valor_de_la_variable

[9] https://www.python.org
[10] Programa informático que permite el acceso a su código de programación.

Cada variable, tiene un nombre y un valor, el cual define a la vez, el tipo de datos de la variable. Existe un tipo de "variable", denominada constante, la cual se utiliza para definir valores fijos, que no requieran ser modificados.

7.3.1.2. Tipos de Datos

Una variable o constante puede tener valores de diversos tipos. Entre ellos tenemos los que se muestran en la Tabla 1.

Tipo	Clase	Notas	Ejemplo
str	Cadena	Inmutable	"Hola"
unicode	Cadena	Versión Unicode de str	u"Hola"
list	Secuencia	Mutable, contiene objetos de diverso tipo	[4,"Hola",3,14]
tuple	Secuencia	Inmutable, contiene objetos de diverso tipo	(4,"Hola",3,14)
set	Conjunto	Mutable, sin orden y sin duplicados	set([4,"Hola",3,14])
frozenset	Conjunto	Inmutable, sin orden, sin duplicados	frozenset([4,"Hola",3,14])
dict	Diccionario	Pares de clave:valor	{"clave1":4, "clave1":"Hola"}
int	Entero	Precisión fija, convierte a long si es necesario	32
long	Entero	Presición arbitraria	32L ó 1298918298398923L
float	Decimal	Coma flotante de doble precisión	3.141592
complex	Complejo	Parte real e imaginaria	(4.5+3j)
bool	Booleano	Valores verdadero o falso	True o False

Tabla 1: Tipos de Datos en Python
Fuente: Ing. Ernesto Freyre G. (2014)

Python, posee además de los tipos ya vistos, 3 tipos más complejos, que admiten una colección de datos. Estos tipos son:

- Tuplas
- Listas
- Diccionarios

Estos tres tipos, pueden almacenar colecciones de datos de diversos tipos y se diferencian por su sintaxis y por la forma en la cual los datos pueden ser manipulados

7.3.1.2.1. Tuplas

Una tupla es una variable que permite almacenar varios datos inmutables[11] de tipos diferentes:

mi_tupla = ('cadena de texto', 15, 2.8, 'otro dato', 25)

Se puede acceder a cada uno de los datos mediante su índice correspondiente, siendo cero, el índice del primer elemento:

print mi_tupla[1] # Salida: 15

También se puede acceder a una porción de la tupla, indicando desde el índice de inicio hasta el índice de fin:

print mi_tupla[1:4] # Devuelve: (15, 2.8, 'otro dato')
print mi_tupla[3:] # Devuelve: ('otro dato', 25)
print mi_tupla[:2] # Devuelve: ('cadena de texto', 15)

Otra forma de acceder a la tupla de forma inversa, es colocando un índice negativo:

print mi_tupla[-1] # Salida: 25
print mi_tupla[-2] # Salida: otro dato

7.3.1.2.2. Listas

Una lista es similar a una tupla con la diferencia fundamental de que permite modificar los datos una vez creados:

mi_lista = ['cadena de texto', 15, 2.8, 'otro dato', 25]

A las listas se accede igual que a las tuplas, por su número de índice:

print mi_lista[1] # Salida: 15
print mi_lista[1:4] # Devuelve: [15, 2.8, 'otro dato']
print mi_lista[-2] # Salida: otro dato

[11] No pueden ser modificados una vez creados.

Las listas no son inmutables: permiten modificar los datos una vez creados:

mi_lista[2] = 3.8 # el tercer elemento ahora es 3.8

Las listas, a diferencia de las tuplas, permiten agregar nuevos valores:

mi_lista.append('Nuevo Dato')

7.3.1.2.3. **Diccionarios**

Mientras que a las listas y tuplas se accede solo y únicamente por un número de índice, los diccionarios permiten utilizar una clave para declarar y acceder a un valor:

mi_diccionario = {'clave_1': valor_1, 'clave_2': valor_2, \'clave_7': valor_7}
print mi_diccionario['clave_2'] # Salida: valor_2

Un diccionario permite eliminar cualquier entrada:

del(mi_diccionario['clave_2'])

Al igual que las listas, el diccionario permite modificar los valores

mi_diccionario['clave_1'] = 'Nuevo Valor'

7.3.1.3. Operadores Aritméticos

Entre los operadores aritméticos que Python utiliza, podemos observar en la Figura 5:

Símbolo	Significado	Ejemplo	Resultado
+	Suma	a = 10 + 5	a es 15
-	Resta	a = 12 - 7	a es 5
-	Negación	a = -5	a es -5
*	Multiplicación	a = 7 * 5	a es 35
**	Exponente	a = 2 ** 3	a es 8
/	División	a = 12.5 / 2	a es 6.25
//	División entera	a = 12.5 / 2	a es 6.0
%	Módulo	a = 27 % 4	a es 3

Figura 5: Operadores Aritméticos
Fuente: Bahit Eugenia (2012)

7.3.1.4. Comentarios

Un archivo no solo puede contener código fuente. También puede incluir comentarios[12]. Los comentarios pueden ser de dos tipos: de una sola línea o multi-línea y se expresan como se muestra en la Figura 6:

```
# Esto es un comentario de una sola línea
mi_variable = 15

"""Y este es un comentario
de varias líneas"""
mi_variable = 15

mi_variable = 15  # Este comentario es de una línea también
```

Figura 6: Tipos de Comentarios
Fuente: Bahit Eugenia (2012)

[12] Notas que los programadores deben dejar explicando que hace cada bloque de código.

8. Cronograma

CRONOGRAMA DE AVANCE TESIS DE GRADO

ACTIVIDADES	DURACIÓN EN DÍAS	DEL 28 DE AGOSTO AL 26 DE JUNIO																																					
		AGOSTO				SEPTIEMBRE				OCTUBRE				NOVIEMBRE				DICIEMBRE				FEBRERO				MARZO				ABRIL				MAYO			
		1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
Diseño de Cronograma de Actividades	1																																						
Luvia de ideas	2																																						
Desarrollo de Introducción y Problema	15																																						
Desarrollo de Objetivos e Hipotesis	10																																						
Desarrollo de Justificaciones	3																																						
Desarrollo de Alcances	2																																						
Desarrollo del Marco Teorico	25																																						
Presentación de Avales y Solicitudes	2																																						
Desarrollo del Capitulo III - Marco Aplicativo	50																																						
Planificación	1																																						
Modelado de Gestión	1																																						
Modelado de Datos	7																																						
Modelado de procesos	10																																						
Generación de la aplicación	10																																						
Pruebas de integracion	5																																						
Pruebas de aceptacion	10																																						
Redacción del Capitulo III - Marco Aplicativo	10																																						
Redacción del Capitulo IV - Estado de la Hipótesis	20																																						
Redacción del Capitulo V - Conclusiones y Recomendaciones	5																																						

Figura 5: Cronograma de Actividades
Fuente: Elaboración Propia (2019)

9. Índice Tentativo

1. Capítulo 1. Marco Introductorio.

1.1. Introducción

1.2. Problema

1.2.1. Antecedentes del Problema

1.2.2. Formulación del Problema

1.3. Objetivos

1.3.1. Objetivo General

1.3.2. Objetivos Específicos

1.4. Hipótesis

1.5. Justificaciones

1.5.1. Justificación Social

1.5.2. Justificación Económica

1.5.3. Justificación Científica

1.6. Alcances y Limites

1.6.1. Alcances

1.6.2. Limites

2. Capítulo 2. Marco Teórico

2.1. Ingeniería de Software

2.2. Herramienta CASE

2.2.1. Características de la Herramienta CASE

2.2.2. Uso de Herramientas CASE

2.2.3. Clasificación de las Herramientas CASE

2.2.4. Ventajas y Desventajas de usar Herramientas CASE

2.2.4.1. Ventajas

2.2.4.2. Desventajas

2.3. Pseudocódigo

2.4. Python

2.4.1. Historia

2.4.2. Elementos de Python

2.4.3. Variables

2.4.4. Tipos de Datos

2.4.5. Operadores Aritméticos

2.4.6. Comentarios

3. **Capítulo 3. Marco Aplicativo**

3.1. Análisis de requisitos

3.2. Diseño

3.3. Construcción

3.4. Evaluación

4. **Capítulo 4. Prueba de Hipótesis**

4.1. Demostración de Hipótesis

4.2. Evaluación y análisis de resultados

5. **Capítulo 5. Conclusiones y Recomendaciones**

5.1. Conclusiones

5.2. Recomendaciones

BIBLIOGRAFIA

ANEXOS

10. Bibliografía

Bahit, E. (2012). *Curso: Python para principiantes.* Buenos Aires: Safe Creative.

Downey, A., Elkner, J., & Meyers, C. (2002). *Aprenda a pensar como un Programador.* Massachusetts: Green Tea Press.

Fernandez, A. (2012). *Python 3 al descubierto.* Madrid: Grupo RC.

Flores, J. (2005). *Método de las 6´D UML - Pseudocódigo - Java (Enfoque algorítmico).* Perú: Universidad de San Martín de Porres (USMP).

Garmire, D. (1999). *Lecture Notes on CASE-Tools: Together/J.* Pensilvania: Geunter Teubner.

Instituto Nacional de Estadística e Informática. (1999). *Herramientas CASE.* Peru: Oficina de Impresiones de la Oficina Técnica de Difusión Estadística y Tecnología Informática del Instituto Nacional de Estadística e Informática (INEI).

Joyanes Aguilar, L. (2008). *Fundamentos de Programación Algoritmos, estructura de datos y objetos* (Cuarta ed.). Madrid: McGraw-Hill/Interamericana de España, S.A.U.

Kendall, K., & Kendall, J. (2011). *Analisis y diseño de Sistemas* (Octava ed.). Mexico: Pearson Educacion.

Letelier, P., & Sánchez, E. (2003). *Metodologías Ágiles en el Desarrollo de Software.* Alicante: Grupo ISSI.

Marti, N., Ortega, Y., & Lopez, J. A. (2004). *Estructuras de datos y métodos algoritmicos: ejercicios resueltos.* Pearson Educación.

Marzal, A., & Gracia, I. (2009). *Introducción a la programación con Python.* España: Sapientia.

Pinales, F. J., & Velázquez, C. E. (2014). *Algoritmos Resueltos con diagramas de flujo y Pseudocódigo* (Primera ed.). Mexico: Departamento Editorial de la Dirección General de Difusión y Vinculación.

Pressman, R. (2005). *Ingeniería del Software Un enfoque Practico.* Mexico: McGraw-Hill.

Rossum, G. V. (2017). *El tutorial de Python.* Argentina: Fred L.

Severance, C. (2016). *Python para informáticos.* Mexico: O'Reilly.

Sommerville, I. (2005). *Ingeniería de Software* (Septima ed.). Madrid: Pearson Educación.