SOLUZIONINNOVATIVE S.A.S.

INSTITUTO SUPERIOR TECNOLÓGICO MARIANO SAMANIEGO

Guía Didáctica de Programación Avanzada

2024

Juan Pardo

SOLUZIONINNOVATIVE S.A.S.

EDITORIAL

Guía Didáctica de Programación Avanzada

ISBN: 978-9942-7294-2-2

Autor:
Juan Pardo

SOLUZIONINNOVATIVE S.A.S.

EDITORIAL

Primera Edición, septiembre 2024

Guía Didáctica de Programación Avanzada

ISBN: 978-9942-7294-2-2

Editado por:

Sello editorial: ©Soluzioninnovative S.A.S. Editorial

No Radicación: 164771

Editorial: ©Soluzioninnovative S.A.S.

Editorial Los Andes y El Sufragio

Dirección de Publicaciones Científicas Soluzioninnovative S.A.S.

Editorial Riobamba, Chimborazo, Ecuador

Teléfono: +593967468602

Código Postal: 060108

https://orcid.org/0009-0009-4690-5589

https://doi.org/10.61396/editorialsoluzioninnovative.lib20

Índice general

Índice general 8

Índice de tablas 11

Índice de figuras 11

Introducción 12

Lineamientos generales 13

Competencias genéricas 13

Competencias específicas 13

Bibliografía 13

Básicas 13

Complementarias 14

Webgrafías 14

PRIMER BIMESTRE 15

1. Unidad I: El camino del programa. 15

1.1. Lenguaje de programación Python. 15

1.2. Lenguajes formales y lenguajes naturales. 16

1.3. Entorno de desarrollo 17

1.4. Primer programa 18

1.5. Ejemplos y/o prácticas 19

1.6. Autoevaluación 1 19

2. Unidad 2: Estructuras de Python 21

2.1. Variables, expresiones y sentencias 21

2.1.1. Variables 21

2.1.2. Estructuras de interconexiones 22

2.1.3. Expresiones 23

2.1.4. Operadores 24

2.2. Funciones 24

2.3. Condiciones y recursividad 26

2.4. Iteraciones 27

2.4.1. Sentencia while (mientras) 27

2.4.2. Uso de else en while 27

2.4.3. Instrucción break 27

2.4.4. Instrucción continue 28

2.4.5. Ejemplo menú interactivo 28

2.4.6. Sentencia for (para) 29

2.5. Cadenas, listas y tuplas 29

2.5.1. Cadenas 29

2.5.2. Listas. 30

2.5.3. Tuplas 30

2.6. Ejemplos y/o prácticas 30

2.7. Autoevaluación 2 31

SEGUNDO BIMESTRE 33

3. Unidad 3: Elementos funcionales de una interfaz gráfica 33

3.1. Conceptos básicos sobre los sistemas de memoria 33

3.1.1. Herencia 33

3.1.2. Herencia simple 33

3.1.3. Herencia múltiple 34

3.2. Utilización de Tkinter 35

3.2.1. Creación de Widget 36

3.3. Flujo de interfaces de menú 38

3.3.1. Menús 38

3.4. Ejemplos y/o prácticas 40

3.5. Autoevaluación 3 41

4. Unidad 4: El modelo de acceso a datos 43

4.1. Conexión con base de datos 43

4.2. Seguridad conexión – desconexión del sistema 43

4.2.1. Pasos para conectarse a una base de datos 43

4.3. CRUD de datos 44

4.4. Diseño de la arquitectura 46

4.4.1. Framework de gran utilidad 46

4.5. Ejemplos y/o prácticas 46

4.6. Autoevaluación 4 47

5. Unidad 5: Informes 49

5.1. Reportes 49

5.1.1. Líneas geométricas 49

5.1.2. Grilla 50

5.2. Ejemplos y/o prácticas 52

5.3. Autoevaluación 5 52

Solucionarios 54

Índice de tablas

Tabla 1: Palabra reservadas 22

Tabla 2: Posiciones en ventana 38

Índice de figuras

Figura 1: Funcionalidad del interprete 15

Figura 2: Funcionalidad el compilador 16

Figura 3: Instalar Python 18

Figura 4. Presentar datos por pantalla 18

Figura 5: Declaración de variables en Python 21

Figura 6: Tipos de datos y variables 22

Figura 7: Operadores en Python 24

Figura 8: Diagrama de clases 33

Figura 9: Primera ventana en Tkinter 36

Figura 10: Widget Tkinter 37

Introducción

El estudio de programación es trascendental durante su formación académica de un Tecnólogo en Desarrollo de Software, conlleva a transformar tareas rutinarias a tareas automatizadas; las mismas que se pueden lograr a través del maravilloso mundo de la programación.

Así como en la construcción de un edificio se realizan planos previos, en el Desarrollo de Software o aplicaciones se deben realizar diagramas en UML[1] para tener una perspectiva completa del comportamiento del software o aplicación a desarrollar, con el propósito que se persigue con la asignatura es desarrollar las competencias básicas y necesarias para que el alumno esté en capacidad de analizar y comprender problemas que involucren temas de software para que pueda resolverlos utilizando UML (clases, secuencia) y la programación asociada a ello; además, elabore e investigue código fuente reutilizable, de fácil uso y sobre todo de mantenimiento sencillo que acoplado a los estándares de programación hagan fácil la comunicación entre programadores de diversos lenguajes.

La asignatura de Programación Avanzada contempla cinco unidades las mismas que se distribuyen en dos bimestres de estudio. En el primer bimestre se estudiará: Estructuras de Control, Programación Orientada a Objetos e Interfaz Gráfica de Usuario y finalmente en el segundo bimestre se revisará: Expresiones Regulares que se usan en Python, Conexiones a Bases de Datos a través de JDBC; ODBC y Reportes

El reto está planteado, los resultados de su formación personal y profesional depende de cuanto esté dispuesto a dar en esfuerzo y responsabilidad. Le invitamos a que juntos llevemos adelante el estudio de la materia y como profesor de la asignatura estoy dispuesto a ser parte de su proceso de enseñanza - aprendizaje, recordándole que esta materia es una de las bases para ir perfilando su proyecto de vida en el ámbito estudiantil y profesional.

[1] Lenguaje Unificado de Modelado

Lineamientos generales

Competencias genéricas

Ensamblar, configurar e interconectar equipos de cómputo aplicando las normas y especificaciones técnicas.

Satisfacer las necesidades de información de las organizaciones mediante la integración de soluciones de tecnologías de la Información y las comunicaciones en procesos empresariales.

Competencias específicas

Trabajo en equipo. - Los estudiantes aprenderán a realizar trabajos de manera colaborativa y cooperativa que será de gran ayuda en la cotidianidad de la vida.

Solución de problemas. - Los estudiantes buscarán la solución a los diferentes problemas que se suscite en el trascurso del estudio de la presente materia, dando una solución oportuna y fiable.

Creatividad. - Los estudiantes desarrollarán los Soft skills mismas que les ayudarán a desarrollar sus capacidades importantes para desempeñar trabajos tanto en el presente como en el futuro

Bibliografía

Básicas

Downey, A., Elkner, J., y Meyers, C. (2002). Aprenda a pensar como un programador con Python. Aprenda a Pensar Como Programador Con Python,

Marzal, A., Llorens, D., & Gracia, I. (2003). Aprender a programar con Python: una experiencia docente. Universitat Jaume I

Pardo-Montero, Juan Pablo. (2023). Guía de Programación Avanzada Python.

Pardo-Montero, Juan Pablo. (2023). Programación Avanzada Python. Programa de estudio de asignatura

Complementarias

García, Á. (2011). Introducción a Python. Autoedición.

Marzal, A., García, I., & García, P. (2014). Introducción a la programación con Python 3. Catellón de la Plana. España: Publicacions de la Universitat Jaume I.Servei de Comunicació i PublicacionsCampus del Riu Sec. Edifici Rectorat i Serveis Centrals. 12071 Castelló de la Plana.

Python, R. (s.f.). Recursos Python. Obtenido de Crear documentos PDF en Python con ReportLab: https://recursospython.com/guias-y-manuales/crear-documentos-pdf-en-python-con-reportlab

Webgrafías

Anónimo. (s.f.). ReportLab. Obtenido de ReportLab Documentation: https://docs.reportlab.com/

Covantec. (2014-2018). Covantec . Programación en Python 3 - Nivel básico https://entrenamiento-python-basico.readthedocs.io/es/3.7/

Documentación Python. (2022). Tutorial de Python—Documentación de Python—3.11.0a6. https://docs.python.org/3.11/tutorial/index.html

Foundation, D. S. (2005). Django. Obtenido de Django documentation : https://docs.djangoproject.com/en/3.0/

Foundation, P. S. (s.f.). Python. Obtenido de Python 3. Documentation: https://docs.python.org/3/

Foundation., P. S. (2001-2019). Python. Obtenido de Python interface to Tcl/Tk: https://docs.python.org/3.5/library/tkinter.html

Foundation., P. S. (s.f.). Python. Obtenido de Python 2. Documentation : https://docs.python.org/2/

PRIMER BIMESTRE

1. Unidad I: El camino del programa.

Para desarrollar los ejercicios se tiene que pensar en diferentes soluciones de una manera creativa y con una buena precisión. El proceso de aprender a programar se necesita desarrollar habilidades blandas, las que permiten desarrollar las soluciones más apropiadas en el desarrollo de los ejercicios planteados

1.1. Lenguaje de programación Python.

Python es un ejemplo de un lenguaje de alto nivel y de propósito general que se destaca por su sintaxis simple y legible, lo que facilita su uso tanto para principiantes como para programadores experimentados. (Vasallo, 2021).

> Inicie con el estudio del texto básico: Aprenda a Pensar Como un Programador con Python Downey, A., Elkner, J., & Meyers, C. (2002). Capítulo 1, sección 1.1 con el tema Lenguaje de Programación Python.

Existen programas que se encargar de convertir el resultado de la codificación: intérpretes y compiladores

- **Intérprete. -** Lee un programa de alto nivel y luego lo ejecuta, lo que significa que hace lo que dice el programa. Poco a poco traduce el programa, leyendo y ejecutando cada comando.

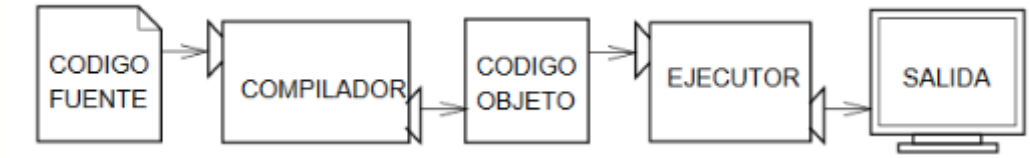
Figura 1

Figura 1: Funcionalidad del interprete

Fuente: (Downey et al., 2002)

- **Compilador. -** Antes de ejecutar cualquiera de las instrucciones, lea el programa y lo traduzca al mismo tiempo. **Figura 2**

Figura 2: Funcionalidad el compilador



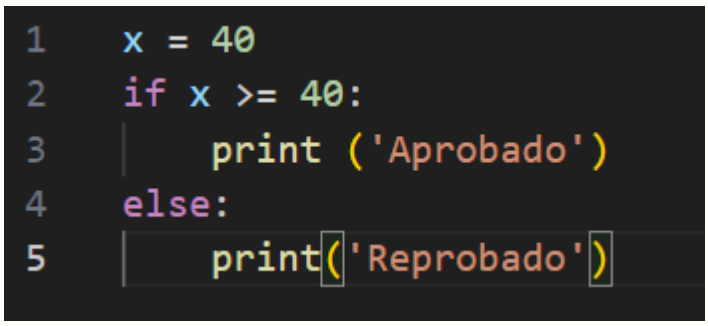
Fuente: (Downey et al., 2002)

1.2. Lenguajes formales y lenguajes naturales.

Según Downey et al., (2022) "Los lenguajes formales son lenguajes creados por humanos que tienen usos particulares. Por ejemplo, la notación matemática es un lenguaje formal porque se utiliza para representar las relaciones entre números y símbolos." (p. 6)

En cambio, los lenguajes naturales son que los seres humanos nos comunicamos como, por ejemplo: español, inglés, francés, etc.

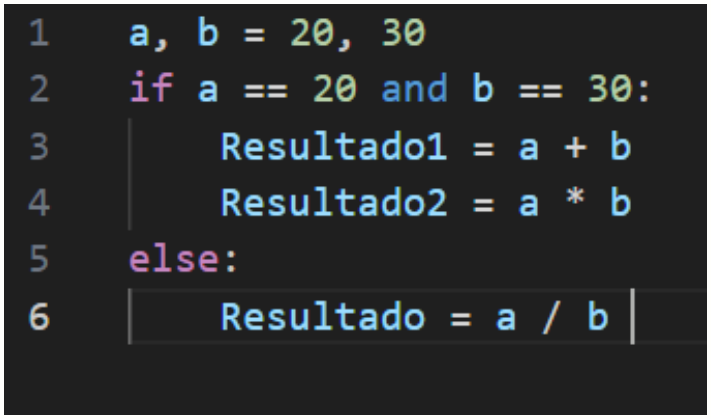
Ahora retome el texto básico: Aprenda a Pensar Como un Programador con Python Downey et al., (2022). Capítulo 1, sección 1.4 con el tema Lenguaje formales y lenguajes naturales

Lenguaje Natural	**Usando programación en Python**
En un instante de tiempo la variable X que corresponde a la nota de un estudiante es igual a 40. Si el valor de la variable "X" es mayor o igual a 40 entonces el estudiante aprueba la asignatura, caso contrario, reprueba la misma.	

```
1   x = 40
2   if x >= 40:
3       print ('Aprobado')
4   else:
5       print('Reprobado')
```

El creador del lenguaje de programación Python creyó conveniente la eliminación de los puntos y comas al final de cada línea de código. Las llaves también se eliminan y para diferenciarlas se utilizan las sangrías.

Si a = 20 y b = 30 entonces realice la suma y la multiplicación y almacénela en una variable resultado1 y resultado2, caso contrario realice la división y almacénela en una variable resultado



```
a, b = 20, 30
if a == 20 and b == 30:
    Resultado1 = a + b
    Resultado2 = a * b
else:
    Resultado = a / b
```

Como se puede observar en el ejercicio anterior, es posible trabajar con dos o más operadores lógicos como OR y AND a través de los cuales se desarrollará la operación especificada. Además es importante mencionar que también se pueden utilizar operadores de comparación como por ejemplo = = , !=, <, >, <=, >=.

1.3. Entorno de desarrollo

Un entorno de desarrollo integrado, también conocido como entorno de desarrollo interactivo (IDE), es un tipo de software que ofrece una variedad de servicios que ayudan al desarrollador o programador a desempeñarse mejor en su trabajo.

En la presente matera de utilizarán los siguientes programas para poseer un entorno de desarrollo óptimo para la utilización de Python:

- **Visual Studio Code.** es un editor de código multiplataforma, ligero y con pocas concesiones a los adornos. Es una herramienta concebida para programar sin distracciones.

  Este editor será el encargado para programar en Python y para que éste ayude a trabajar se tiene que instalar algunos complementos que son:

  - Python snippets
  - Python

- **DB Browser for SQLite:** El presente programa ayuda a tener la administración de una base de datos en el motor SQLite
- **XAMPP:** Ayuda a tener un servidor web, de éste solo se utiliza el programa phpMyAdmin.
- **Python: L**enguaje de programación multiparadigma, soporta distintos modelos de programación. es un lenguaje que se lo puede interpretar y compilar como el programador lo desee.

Cuando se lo instale hay que considerar que le debe adherirlo al path del sistema operativo como muestra en la **Figura 3**

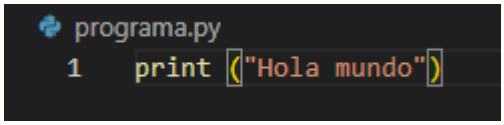
Figura 3: Instalar Python

Fuente: Autoría propia

1.4. Primer programa

Tradicionalmente el primer programa en un lenguaje nuevo se llama\Hola, mundo"(Helloworld!) porque sólo muestra las palabras \Hola a todo el mundo". En Python es así:

Figura 4. Presentar datos por pantalla



Fuente: Autoría propia

Esta es una **Figura 4** se presenta una sentencia print, que no imprime nada en papel, sino que muestra un valor. En esta situación, el resultado son las palabras "Hola, mundo". El valor comienza y termina en las comillas, no en el resultado. Algunas personas evalúan la calidad de un lenguaje de programación por la simplicidad del programa "Hola, mundo". Si aplicamos ese criterio, Python cumple con todos sus objetivos.

Para aprender y conocer de la estructura de Python visite la página web: https://entrenamiento-python-basico.readthedocs.io/es/3.7/leccion1/holamundo.html#ejecucion

1.5. Ejemplos y/o prácticas

Instale en su máquina los diferentes programas, estudiados para poder utilizar Python; además realice ejercicios para familiarizarse con los programas instalados

1.6. Autoevaluación 1

Responda a las siguientes preguntas según corresponda.

1. ¿Cómo funciona un intérprete?
   a) Traduce todo el programa antes de ejecutarlo.
   b) Traduce el programa línea por línea durante la ejecución.
   c) Compila el programa completo y luego lo ejecuta.
   d) Traduce solo las funciones principales antes de la ejecución.
2. ¿Qué es el código fuente?
   a) Es el programa escrito en un lenguaje de alto nivel.
   b) Es el programa en su forma ejecutable.
   c) Es el archivo resultante después de la compilación.
   d) Es el programa en lenguaje de máquina.
3. ¿Cuáles son las formas de usar un intérprete?
   a) Solo en modo comando.
   b) Solo en modo guion.
   c) En modo comando y modo guion.
   d) Solo en modo interactivo.

4. ¿Las instrucciones de los lenguajes de programación son iguales en todos los lenguajes?

a) Sí, todos los lenguajes tienen las mismas instrucciones.

b) No, varían según el lenguaje.

c) Solo en lenguajes de bajo nivel.

d) Dependen del compilador.

5. ¿Qué son los lenguajes formales?

a) Son lenguajes hablados diseñados para la comunicación cotidiana.

b) Son lenguajes diseñados para fines específicos, como la programación.

c) Son lenguajes usados solo en inteligencia artificial.

d) Son lenguajes que solo se utilizan en matemáticas.

6. ¿Cuáles son las funciones básicas en casi todos los lenguajes de programación?

a) Entrada, salida, operaciones matemáticas, condicionales y bucles.

b) Entrada, manejo de gráficos y seguridad de datos.

c) Entrada, acceso a hardware y manipulación de memoria.

d) Entrada, operaciones con redes y manejo de bases de datos.

2. Unidad 2: Estructuras de Python

Una vez estudiado y conocido el funcionamiento de Python, en la presente unidad nos centraremos en la construcción de programas utilizando el lenguaje de estudio. Para lo cual, los invito a recordar conceptos anteriores de programación porque el objetivo de esta materia es aprender un nuevo lenguaje de programación.

2.1. Variables, expresiones y sentencias

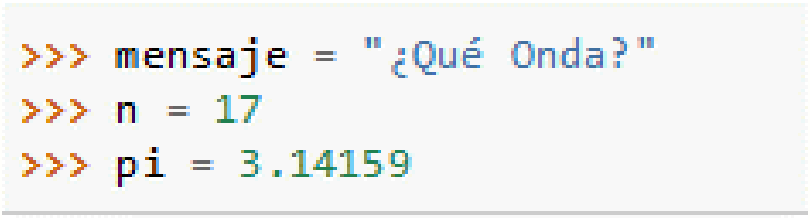
A continuación, conoceremos cual es el proceso para declarar variables, el manejo de expresiones y familiarizarse con las sentencian en Python

> Ahora retome el texto básico: Aprenda a Pensar Como un Programador con Python Downey, A., Elkner, J., & Meyers, C. (2002). Capítulo 2: Variables, expresiones y sentencias.

2.1.1. Variables

Huet Pablo (2024) describe las variables en Python como "cajas mágicas" que permiten almacenar una variedad de datos sin necesidad de definir su tipo previamente. La asignación de valores a estas variables se realiza mediante una sentencia específica que crea nuevas variables, brindando flexibilidad en la manipulación de datos. **Figura 5**.

Figura 5: Declaración de variables en Python



Fuente: Pardo-Montero (2023)

Este ejemplo se especifica tres asignaciones. La primera un valor tipo cadena "¿Qué Onda?" a la que se la denominada **mensaje**. A la siguiente de le asigna un valor entero 17 a la variable que se la denominará **n**, y la última se le asigna un valor de punto flotante 3.14159 a la que se denominará **pi**.

Las palabras reservadas que se presentan en la **Tabla 1** no pueden ser utilizadas como nombres de variables

and	del	for	is	raise	assert
if	else	elif	from	lambda	return
break	global	not	try	class	except
or	while	continue	exec	import	yield
def	finally	in	print		

Tabla 1: Palabra reservadas

Fuente: Pardo-Montero (2023)

Figura 6: Tipos de datos y variables

Fuente: Curso Píldoras informáticas

2.1.2. Estructuras de interconexiones

Se refieren a las formas en que los diferentes componentes del lenguaje y sus bibliotecas se conectan y colaboran entre sí. Una sentencia en Python es una instrucción o asignación que permite al programa interactuar con los usuarios y realizar las operaciones solicitadas por ellos.

Por ejemplo

```
print 1
x = 2
print x
```

Produce la salida:

```
1
2
```

La sentencia asignada no causa salida deseada.

2.1.3. Expresiones

Según la Real Academia Española (s.f.), la expresión “Conjunto de números y símbolos ligados entre sí por los signos de una operación” se puede manifestar una expresión incluye valores, variables y operadores. Una ventaja de Python es que permite ejecutar estas expresiones directamente en la línea de comandos; el intérprete procesará y mostrará los resultados en pantalla, como se ilustra en el siguiente ejemplo:

```
>>> 1 + 1
2
```

Las expresiones aparecen en el lado derecho de las sentencias de asignación porque producen un valor al evaluar una expresión. Se considera un valor como una expresión, y lo mismo se aplica a las variables declaradas.

```
>>> 17
17
>>> x
2
```

Evaluar una expresión no es lo mismo que mostrar valor en el lenguaje natural de las personas. Es confuso.

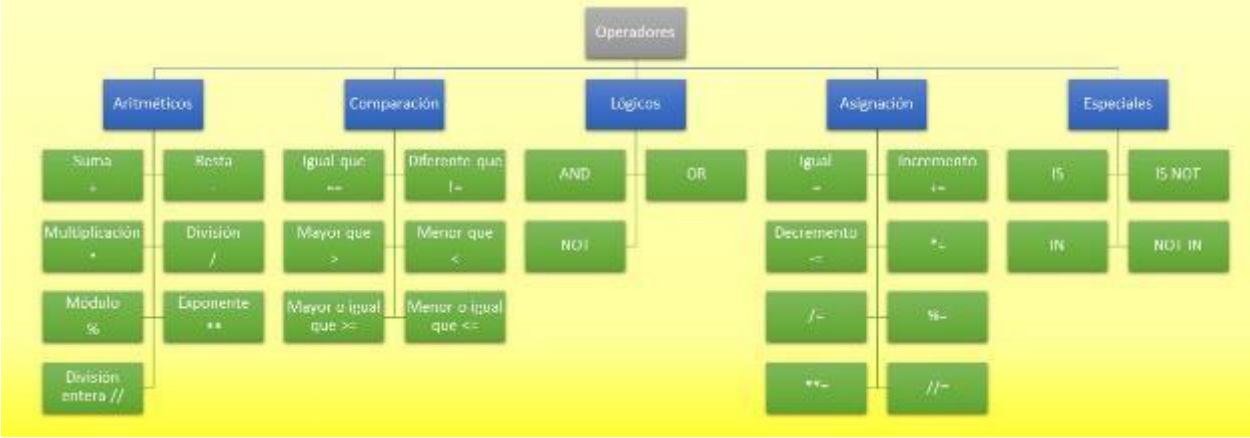
```
>>> mesaje = "Juan es chido"
>>> mesaje
'Juan es chido'
>>> print mesaje
Juan es chido
```

Para recibir el valor, la terminal de Python usa el mismo formato cuando muestra la expresión que ha evaluado. Esto significa que las cadenas incluyen comillas. Sin embargo, el valor de la expresión, que en este caso es el contenido de la cadena, se despliega por la sentencia print

2.1.4. Operadores

Los símbolos especiales utilizados para representar cómputos como la suma y la multiplicación se conocen como operadores. Los valores utilizados por el operador se denominan operandos **Figura 7**.

Figura 7: Operadores en Python



Fuente: Curso de píldoras informáticas

2.2. Funciones

Según Huet Pablo (2024), son bloques de código organizados y reutilizables que están diseñados para realizar una tarea específica. Está asociado a un nombre y recibe uno o más argumentos como

Ahora retome el texto básico: Aprenda a Pensar Como un Programador con Python Downey, A., Elkner, J., & Meyers, C. (2002). Capítulo 3: Funciones

entrada y sigue una secuencia de sentencias que ejecutan una operación deseada, devuelven un valor y/o realizan una tarea. Este bloque puede ser llamado cuando sea necesario.

Leído el apartado de funciones usted está en la capacidad de construir una función y llamarla cuando el programa lo necesite, expongo realizar un conjunto de ejercicios que se presentan a continuación:

- Realiza una función llamada **área_rectángulo (base, altura)** que devuelva el área del rectángulo a partir de una base y una altura. Calcula el área de un rectángulo de 15 de base y 10 de altura:

```
def area_rectangulo(base, altura):
    return base*altura

print( area_rectangulo(15,10) )
```

```
150
```

- Realiza una función llamada **relación (a, b)** que a partir de dos números cumpla lo siguiente:
  - Si el primer número es mayor que el segundo, debe devolver 1.
  - Si el primer número es menor que el segundo, debe devolver -1.
  - Si ambos números son iguales, debe devolver un 0.

Comprueba la relación entre los números: '5 y 10', '10 y 5' y '5 y 5'.

```
def relacion(a, b):
    if a > b:
        return 1
    elif a < b:
        return -1
    else:
        return 0

print( relacion(5, 10) )
print( relacion(10, 5) )
print( relacion(5, 5) )
```

```
-1
1
0
```

2.3. Condiciones y recursividad

Son funciones que se llaman a sí mismas durante su propia ejecución. Ellas funcionan de forma similar a las iteraciones, pero debe encargarse de planificar el momento en que dejan de llamarse a sí mismas o tendrá una función recursiva infinita.

Ahora retome el texto básico: Aprenda a Pensar Como un Programador con Python Downey, A., Elkner, J., & Meyers, C. (2002). Capítulo 4: Condiciones y Recursividad

- Un ejemplo de una función recursiva sin retorno es el ejemplo de cuenta regresiva hasta cero a partir de un número:

```
>>> def cuenta_regresiva(numero):
...     numero -= 1
...     if numero > 0:
...         print numero
...         cuenta_regresiva(numero)
...     else:
...         print "Booooooooom!"
...     print "Fin de la función", numero
...
>>> cuenta_regresiva(5)
4
3
2
1
Booooooooom!
Fin de la función 0
Fin de la función 1
Fin de la función 2
Fin de la función 3
Fin de la función 4
```

- Un ejemplo de una función recursiva con retorno es el ejemplo del cálculo de la factorial de un número corresponde al producto de todos los números desde 1 hasta el propio número. Es el ejemplo con retorno más utilizado para mostrar la utilidad de este tipo de funciones:

```
>>> def factorial(numero):
...     print "Valor inicial ->",numero
...     if numero > 1:
...         numero = numero * factorial(numer
...     print "valor final ->",numero
...     return numero
...
>>> print factorial(5)
Valor inicial -> 5
Valor inicial -> 4
Valor inicial -> 3
Valor inicial -> 2
Valor inicial -> 1
valor final -> 1
valor final -> 2
valor final -> 6
valor final -> 24
valor final -> 120
120
```

2.4. Iteraciones

Se denominará iteración a aquel conjunto de instrucciones que se encuentran acotadas por un bloque repetido de acciones indicadas a través de comandos en el lenguaje Python. Las instrucciones por excelencia en este lenguaje pertenecen a los bloques **for** y **while**. Estos bloques poseen Indentación para reconocer el bloque como tal.

> Ahora retome el texto básico: Aprenda a Pensar Como un Programador con Python Downey, A., Elkner, J., & Meyers, C. (2002). Capítulo 6: Iteraciones.

2.4.1. Sentencia while (mientras)

Se basa en evaluar una condición lógica, siempre que sea verdadera, y repetir un bloque. El programador debe elegir cuándo la condición debe cambiar a Falso para que el While finalice.

```
c = 0
while c <= 5:
    c+=1
    print("c vale", c)
```

```
c vale 1
c vale 2
c vale 3
c vale 4
c vale 5
c vale 6
```

2.4.2. Uso de else en while

Para ejecutar un bloque de código una vez que la condición ya no devuelva True (normalmente al final), se encadena al While:

```
c = 0
while c <= 5:
    c+=1
    print("c vale", c)
else:
    print("Se ha completado toda la iteración y c vale", c)
```

```
c vale 1
c vale 2
c vale 3
c vale 4
c vale 5
c vale 6
Se ha completado toda la iteración y c vale 6
```

2.4.3. Instrucción break

En cualquier momento, puede "romper" la ejecución del While. El Else solo se llama al finalizar la iteración, por lo que no se ejecutará.

```
c = 0
while c <= 5:
    c+=1
    if (c==4):
        print("Rompemos el bucle cuando c vale", c)
        break
    print("c vale",c)
else:
    print("Se ha completado toda la iteración y c vale", c)
```

```
c vale 1
c vale 2
c vale 3
Rompemos el bucle cuando c vale 4
```

2.4.4. Instrucción continue

Utilice este método para "saltar" la iteración actual sin romper el ciclo.

```
c = 0
while c <= 5:
    c+=1
    if c==3 or c==4:
        # print("Continuamos con la siguiente iteración", c)
        continue
    print("c vale",c)
else:
    print("Se ha completado toda la iteración y c vale", c)
```

```
c vale 1
c vale 2
c vale 5
c vale 6
Se ha completado toda la iteración y c vale 6
```

2.4.5. Ejemplo menú interactivo

```
print("Bienvenido al menú interactivo")
while(True):
    print("""¿Qué quieres hacer? Escribe una opción
    1) Saludar
    2) Sumar dos números
    3) Salir""")
    opcion = input()
    if opcion == '1':
        print("Hola, espero que te lo estés pasando bien")
    elif opcion == '2':
        n1 = float(input("Introduce el primer número: "))
        n2 = float(input("Introduce el segundo número: "))
        print("El resultado de la suma es: ",n1+n2)
    elif opcion =='3':
        print("¡Hasta luego! Ha sido un placer ayudarte")
        break
    else:
        print("Comando desconocido, vuelve a intentarlo")
```

```
Bienvenido al menú interactivo
¿Qué quieres hacer? Escribe una opción
    1) Saludar
    2) Sumar dos números
    3) Salir
1
Hola, espero que te lo estés pasando bien
¿Qué quieres hacer? Escribe una opción
    1) Saludar
    2) Sumar dos números
    3) Salir
2
Introduce el primer número: 10
Introduce el segundo número: 5
El resultado de la suma es:  15.0
¿Qué quieres hacer? Escribe una opción
    1) Saludar
    2) Sumar dos números
    3) Salir
kdjsk
Comando desconocido, vuelve a intentarlo
¿Qué quieres hacer? Escribe una opción
    1) Saludar
    2) Sumar dos números
    3) Salir
3
¡Hasta luego! Ha sido un placer ayudarte
```

2.4.6. Sentencia for (para)

Permite ejecutar una instrucción o un conjunto de instrucciones varias veces, según el número de repeticiones especificado.

```
numeros = [1,2,3,4,5,6,7,8,9,10]
indice = 0

for numero in numeros:  # Para [variable] en [lista]
    print(numero)
```

```
1
2
3
4
5
6
7
8
9
10
```

2.5. Cadenas, listas y tuplas

Las cadenas, listas y tuplas son distintos tipos de secuencias. Una secuencia es un tipo de objeto que almacena datos y que permite el acceso a una parte determinada de su información utilizando índices.

Las listas, tuplas, diccionarios y conjuntos (set) son estructuras que permiten trabajar con colecciones de datos. El primer elemento de una lista o de una tupla ocupa la posición 0.

Ahora retome el texto básico: Aprenda a Pensar Como un Programador con Python Downey, A., Elkner, J., & Meyers, C. (2002). Capítulo 7,8 y 9: Cadenas listas y tuplas

2.5.1. Cadenas

Las cadenas de caracteres, comúnmente conocidas como strings, son un tipo de dato que consiste en una secuencia de símbolos como, por ejemplo.

```
>>> 'Hola Mundo'
'Hola Mundo'
>>> "Hola Mundo"
'Hola Mundo'
```

```
>>> u'Jekechitü'
u'Jekechit\xfc'
```

2.5.2. Listas.

Las listas son un tipo de dato compuesto que puede contener diferentes valores, conocidos como ítems o elementos, organizados entre corchetes [ ] y separados por comas como por ejemplo.

```
datos = [4, "Una cadena", -15, 3.14, "Otra cadena"

print(datos[0])
print(datos[-1])
print(datos[2:])
```

```
4
Otra cadena
[-15, 3.14, 'Otra cadena']
```

2.5.3. Tuplas

Son colecciones similares a las listas, pero con la particularidad de que no se pueden modificar. Por ejemplo

```
print(tupla)
print(tupla[0])
print(tupla[-1])
print(tupla[2:])
print(tupla[2][-1])
```

```
(100, 'Hola', [1, 2, 3], -50)
100
-50
([1, 2, 3, 4], -50)
4
```

2.6. Ejemplos y/o prácticas

- Realice una aplicación en Python que permita el ingreso de un número, que tiene que ser positivo caso contrario que pida el ingreso de un nuevo número; con cinco intentos al sexto debe presentar un mensaje que ya excedió el número de intentos permitidos
- Escribe un programa que solicite el ingreso de las notas de 10 parámetros de un estudiante, el programa debe informar si aprueba, supletorio o reprueba
- Reprobados menor a 9
- Supletorios mayores a 9.1 y menores a 13.9
- Aprobados superiores a 14
- En una empresa trabajan N empleados cuyos sueldos oscilan entre $100 y $1000. Realice un programa que informe de cuantos empleados cobran menos de $500 y cuantos más de $500. Informar también del total que gasta la empresa en pagar a sus empleados.
- Realice un programa que imprima 25 términos de la serie 11 – 22 – 33…
- Compruebe que un correo electrónico es correcto utilizando tuplas en Python

2.7. Autoevaluación 2

Responda a las siguientes preguntas según corresponda.

1. ¿Qué caracteriza a un lenguaje de programación como Python?
   a) La capacidad de manipular variables.
   b) La velocidad de ejecución de los programas.
   c) La facilidad para crear interfaces gráficas.
   d) La capacidad de gestionar bases de datos.
2. ¿Cuál es la longitud permitida para los nombres de las variables en Python?
   a) Deben tener una longitud fija de 10 caracteres.
   b) Pueden tener una longitud arbitraria.
   c) Deben ser de 1 a 5 caracteres.
   d) Solo se permiten nombres cortos para evitar errores.
3. ¿Qué es una sentencia en Python?
   a) Una instrucción que puede ser ejecutada por el intérprete de Python.
   b) Un tipo de dato que se declara al principio del código.
   c) Un bloque de código que se ejecuta en paralelo.
   d) Un nombre reservado para funciones especiales.
4. ¿Qué es el resultado de una sentencia print en Python?
   a) Una expresión.
   b) Un valor numérico.
   c) Una cadena de texto que se muestra en la consola.
   d) Un tipo de variable.
5. ¿Qué hace la función type en Python?
   a) Muestra el tipo de un valor o de una variable.
   b) Cambia el tipo de una variable.
   c) Calcula el valor absoluto de un número.
   d) Convierte una cadena de texto en un número.
6. ¿Qué sucede si una recursión no alcanza nunca el caso base?
   a) Seguirá haciendo llamadas recursivas para siempre y nunca terminará.
   b) Terminará automáticamente al alcanzar un límite de recursiones.
   c) Se convertirá en un bucle infinito.
   d) Generará un error de sintaxis.
7. ¿Qué es una porción de cadena?

a) Una parte de una cadena que se extrae usando slicing.

b) La presentación de toda la cadena como una sola unidad.

c) Un tipo de dato que representa múltiples cadenas.

d) Una función que convierte cadenas en números.

8. ¿Cómo se identifican los valores en una lista en Python?

a) Con el nombre de la lista y el índice del valor.

b) Solo con el nombre de la lista.

c) A través de una clave única para cada valor.

d) Usando una etiqueta que describe el valor.

9. ¿Qué es una tupla en Python?

a) Un conjunto de valores agrupados en una sola estructura.

b) Un conjunto de cadenas agrupadas.

c) Una lista ordenada de valores mutables.

d) Una función que agrupa cadenas en una lista.

SEGUNDO BIMESTRE

3. Unidad 3: Elementos funcionales de una interfaz gráfica

En la presente unidad abordaremos diferentes librerías para poder crear interfaces gráficas en entorno de escritorio.

3.1. Conceptos básicos sobre los sistemas de memoria

La programación orientada a objetos (POO, también conocida como OOP) es un paradigma de programación que revoluciona cómo obtener resultados. Los objetos manipulan los datos de entrada para producir datos de salida específicos, donde cada objeto proporciona una función única.

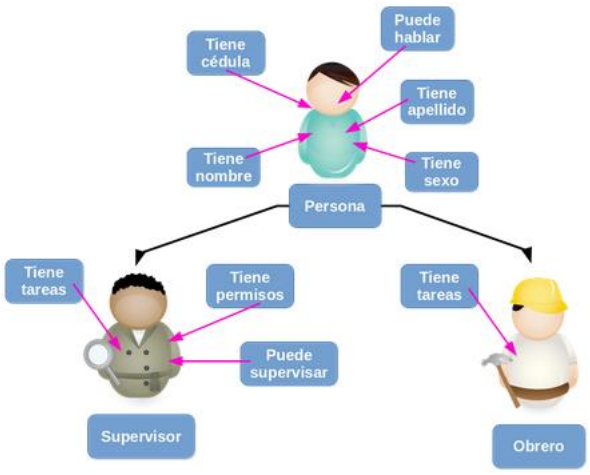
> Para recordar los pilares de la Programación Orientada a Objetos visite la página web: https://entrenamiento-python-basico.readthedocs.io/es/3.7/leccion9/poo.html

3.1.1. Herencia

La herencia permite que una clase adquiera atributos y métodos de otra, lo que favorece la reutilización del código y facilita la creación de jerarquías de clases. En Python, al igual que en otros lenguajes de programación, la herencia se puede implementar. Sin embargo, Python también admite la herencia múltiple, lo que significa que una clase puede derivar de varias clases base al mismo tiempo.

3.1.2. Herencia simple

Figura 8: Diagrama de clases



Fuente: autoría propia

```python
class Persona(object):
    """Clase que representa una Persona"""

    def __init__(self, cedula, nombre, apellido, sexo):
        """Constructor de clase Persona"""
        self.cedula = cedula
        self.nombre = nombre
        self.apellido = apellido
        self.sexo = sexo

    def __str__(self):
        """Devuelve una cadena representativa de Persona"""
        return "%s: %s, %s %s, %s." % (
            self.__doc__[25:34], str(self.cedula), self.nombre,
            self.apellido, self.getGenero(self.sexo))

    def hablar(self, mensaje):
        """Mostrar mensaje de saludo de Persona"""
        return mensaje

    def getGenero(self, sexo):
        """Mostrar el genero de la Persona"""
        genero = ('Masculino','Femenino')
        if sexo == "M":
            return genero[0]
        elif sexo == "F":
            return genero[1]
        else:
            return "Desconocido"
```

```python
class Supervisor(Persona):
    """Clase que representa a un Supervisor"""

    def __init__(self, cedula, nombre, apellido, sexo, rol):
        """Constructor de clase Supervisor"""

        # Invoca al constructor de clase Persona
        Persona.__init__(self, cedula, nombre, apellido, sexo)

        # Nuevos atributos
        self.rol = rol
        self.tareas = ['10','11','12','13']

    def __str__(self):
        """Devuelve una cadena representativa al Supervisor"""
        return "%s: %s %s, rol: '%s', sus tareas: %s." % (
            self.__doc__[26:37], self.nombre, self.apellido,
            self.rol, self.consulta_tareas())

    def consulta_tareas(self):
        """Mostrar las tareas del Supervisor"""
        return ', '.join(self.tareas)
```

3.1.3. Herencia múltiple

```python
class Destreza(object):
    """Clase la cual representa la Destreza de la Persona"""

    def __init__(self, area, herramienta, experiencia):
        """Constructor de clase Destreza"""
        self.area = area
        self.herramienta = herramienta
        self.experiencia = experiencia

    def __str__(self):
        """Devuelve una cadena representativa de la Destreza"""
        return """Destreza en el área %s con la herramienta %s,
        tiene %s años de experiencia.""" % (
            str(self.area), self.experiencia, self.herramienta)


class JefeCuadrilla(Supervisor, Destreza):
    """Clase la cual representa al Jefe de Cuadrilla"""

    def __init__(self, cedula, nombre, apellido, sexo,
        rol, area, herramienta, experiencia, cuadrilla):
        """Constructor de clase Jefe de Cuadrilla"""

        # Invoca al constructor de clase Supervisor
        Supervisor.__init__(self, cedula, nombre, apellido, sexo,
            rol)
        # Invoca al constructor de clase Destreza
        Destreza.__init__(self, area, herramienta, experiencia)

        # Nuevos atributos
        self.cuadrilla = cuadrilla
```

Ahora retome el texto básico: Aprenda a Pensar Como un Programador con Python Downey, A., Elkner, J., & Meyers, C. (2002). Capítulo 16: Herencia

```python
class Persona(object):
    """Clase que representa una Persona"""

    def __init__(self, cedula, nombre, apellido, sexo):
        """Constructor de clase Persona"""
        self.cedula = cedula
        self.nombre = nombre
        self.apellido = apellido
        self.sexo = sexo

    def __str__(self):
        """Devuelve una cadena representativa de Persona"""
        return "%s: %s, %s %s, %s." % (
            self.__doc__[25:34], str(self.cedula), self.nombre,
            self.apellido, self.getGenero(self.sexo))

    def hablar(self, mensaje):
        """Mostrar mensaje de saludo de Persona"""
        return mensaje

    def getGenero(self, sexo):
        """Mostrar el genero de la Persona"""
        genero = ('Masculino','Femenino')
        if sexo == "M":
            return genero[0]
        elif sexo == "F":
            return genero[1]
        else:
            return "Desconocido"
```

```python
class Supervisor(Persona):
    """Clase que representa a un Supervisor"""

    def __init__(self, cedula, nombre, apellido, sexo, rol):
        """Constructor de clase Supervisor"""

        # Invoca al constructor de clase Persona
        Persona.__init__(self, cedula, nombre, apellido, sexo)

        # Nuevos atributos
        self.rol = rol
        self.tareas = ['10','11','12','13']

    def __str__(self):
        """Devuelve una cadena representativa al Supervisor"""
        return "%s: %s %s, rol: '%s', sus tareas: %s." % (
            self.__doc__[26:37], self.nombre, self.apellido,
            self.rol, self.consulta_tareas())

    def consulta_tareas(self):
        """Mostrar las tareas del Supervisor"""
        return ', '.join(self.tareas)
```

La sobrecarga de métodos es también es conocida por Overriding Methods, le permite sustituir un método proveniente de la Clase Base, en la Clase Derivada debe definir un método con la misma forma (es decir, mismo nombre de método y mismo número de parámetros que como está definido en la Clase Base).

```
>>> class Persona():
...     def __init__(self):
...         self.cedula = 13765890
...     def mensaje(self):
...         print("mensaje desde la clase Persona")
...
>>> class Obrero(Persona):
...     def __init__(self):
...         self.__especialista = 1
...     def mensaje(self):
...         print("mensaje desde la clase Obrero")
...
>>> obrero_planta = Obrero()
>>> obrero_planta.mensaje()
mensaje desde la clase Obrero
>>>
```

Ahora retome el texto básico: Aprenda a Pensar Como un Programador con Python Downey, A., Elkner, J., & Meyers, C. (2002). Capítulo 14: Clases y Métodos.

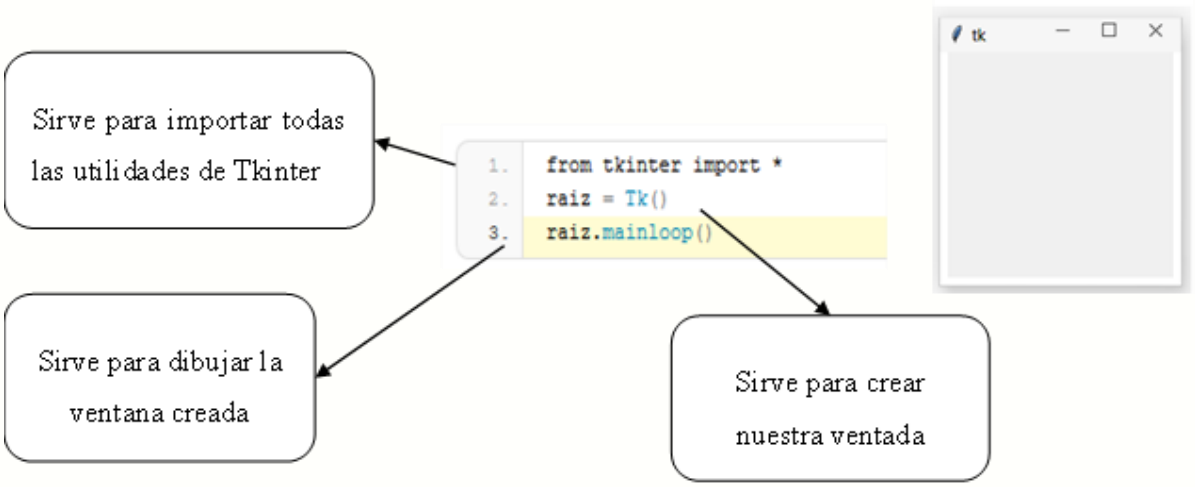
3.2. Utilización de Tkinter

Python ofrece varias librerías para crear interfaces gráficas, entre las más destacadas se encuentran: Tkinter, WxPython, PyQT y PyGTK. En el desarrollo de esta asignatura, utilizaremos Tkinter,

ya que viene incluida con la instalación estándar de Python, lo que facilita su uso para el aprendizaje. Si deseas explorar alguna de las otras librerías mencionadas, será necesario instalarlas previamente.

Según Díaz (2023), Tkinter es un framework GUI integrado en Python que forma parte de la biblioteca estándar, lo que significa que no es necesario instalar dependencias adicionales. Con Tkinter, es posible crear ventanas para desarrollar aplicaciones en la plataforma que elijas. En este caso, trabajaremos en un entorno Windows."

Figura 9: Primera ventana en Tkinter



Fuente: Autoría propia

3.2.1. Creación de Widget

Un widget en Tkinter es un componente gráfico que permite a los usuarios interactuar con una aplicación a través de una interfaz gráfica. Los widgets son los bloques fundamentales que conforman las ventanas de las aplicaciones y pueden ser botones, etiquetas, cuadros de texto, menús, entre otros. Cada widget tiene propiedades que pueden ser configuradas, como su texto, color, tamaño y comportamientos específicos.

Tkinter ofrece una amplia variedad de widgets que permiten crear interfaces gráficas completas, brindando la flexibilidad necesaria para desarrollar desde aplicaciones simples hasta sistemas más complejos. Entre los elementos que se pueden crear se encuentran los que se muestran en la **Figura 10**.

Figura 10: Widget Tkinter

tk	ttk	tix
Button	Button	Ballon
Canvas	Checkbutton	ButtonBox
Checkbutton	Combobox	CheckList
Entry	Entry	ComboBox
Frame	Frame	Control
Label	Label	DialogShell
LabelFrame	LabeledScale	DirList
Listbox	Labelframe	DirSelectBox
Menu	Menubutton	DirSelectDialog
Menubutton	Notebook	DirTree
Message	OptionMenu	ExFileSelectBox
OptionMenu	Panedwindow	ExFileSelectDialog
PanedWindow	Progressbar	FileEntry
Radiobutton	Radiobutton	FileSelectBox
Scale	Scale	FileSelectDialog
Scrollbar	Scrollbar	Hlist
Spinbox	Separator	InputOnly
Text	Sizegrip	LabelEntry
	Treeview	LabelFrame
		ListNoteBook
		Meter
		NoteBook
		OptionMenu
		PanedWindow
		ResizeHandle
		ScrolledHList
		ScrolledListBox
		ScrolledText
		ScrolledTList
		ScrolledWindow
		Select
		Shell
		StdButtonBox
		Tlist
		Tree

Fuente: https://guia-tkinter.readthedocs.io/es/develop/

Para poder utilizar los widgets:

- TK: se utiliza la siguiente línea al inicio del código **from tkinter import***
- TTK: se utiliza la siguiente línea al inicio del código **from tkinter import ttk**

- TTX: se utiliza la siguiente línea al inicio del código **from tkinter import ttx**

3.3. Flujo de interfaces de menú

Para poder posicionar los diferentes widgets en una ventana se imagina que es una matriz con sus diferentes filas y columnas **Tabla 2**.

row =1 , column=1	row =1 , column=2	row =1 , column=3	row =1 , column=4	row =1 , column=5
row =2 , column=1	row =2 , column=2	row =2 , column=3	row =2 , column=4	row =2 , column=5
row =3 , column=1	row =3 , column=2	row =3 , column=3	row =3 , column=4	row =3 , column=5
row =4 , column=1	row =4 , column=2	row =4 , column=3	row =4 , column=4	row =4 , column=5

Tabla 2: Posiciones en ventana

Fuente: https://guia-tkinter.readthedocs.io/es/develop/

3.3.1. Menús

El primer widget de menú que creamos se refiere a la barra de menú, por lo que comúnmente se le denomina menubar.

menu.py

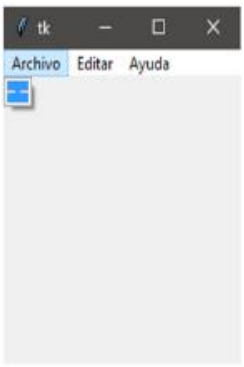
```
from tkinter import *

root = Tk()

menubar = Menu(root)
root.config(menu=menubar)  # Lo asignamos a la base

root.mainloop()
```

Una vez creada la barra, podemos empezar a agregar submenús y comandos. Comencemos con los submenús.

```
filemenu = Menu(menubar)
editmenu = Menu(menubar)
helpmenu = Menu(menubar)
```

Ya tenemos los submenús, pero todavía nos falta añadirlos a la barra de menú

```
menubar.add_cascade(label="Archivo", menu=filemenu)
menubar.add_cascade(label="Editar", menu=editmenu)
menubar.add_cascade(label="Ayuda", menu=helpmenu)
```



Ya tenemos nuestra barra con los tres submenús funcionando correctamente, pero ocurre algo inusual: aparece un elemento predeterminado. Podemos eliminarlo configurando el parámetro tearoff=0.

```
filemenu = Menu(menubar, tearoff=0)
editmenu = Menu(menubar, tearoff=0)
helpmenu = Menu(menubar, tearoff=0)
```

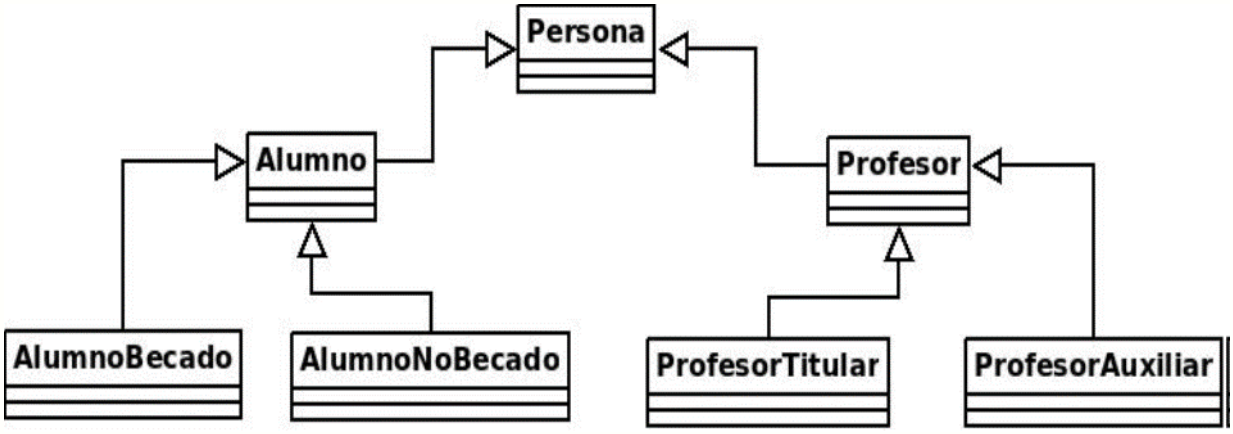
El cófigo completo del menú queda de la siguiente manera:

menu.py

```
from tkinter import *

# Configuración de la raíz
root = Tk()

menubar = Menu(root)
root.config(menu=menubar)

filemenu = Menu(menubar, tearoff=0)
filemenu.add_command(label="Nuevo")
filemenu.add_command(label="Abrir")
filemenu.add_command(label="Guardar")
filemenu.add_command(label="Cerrar")
filemenu.add_separator()
filemenu.add_command(label="Salir", command=root.quit)

editmenu = Menu(menubar, tearoff=0)
editmenu.add_command(label="Cortar")
editmenu.add_command(label="Copiar")
editmenu.add_command(label="Pegar")

helpmenu = Menu(menubar, tearoff=0)
helpmenu.add_command(label="Ayuda")
helpmenu.add_separator()
helpmenu.add_command(label="Acerca de...")

menubar.add_cascade(label="Archivo", menu=filemenu)
menubar.add_cascade(label="Editar", menu=editmenu)
menubar.add_cascade(label="Ayuda", menu=helpmenu)

# Finalmente bucle de la aplicación
root.mainloop()
```

3.4. Ejemplos y/o prácticas

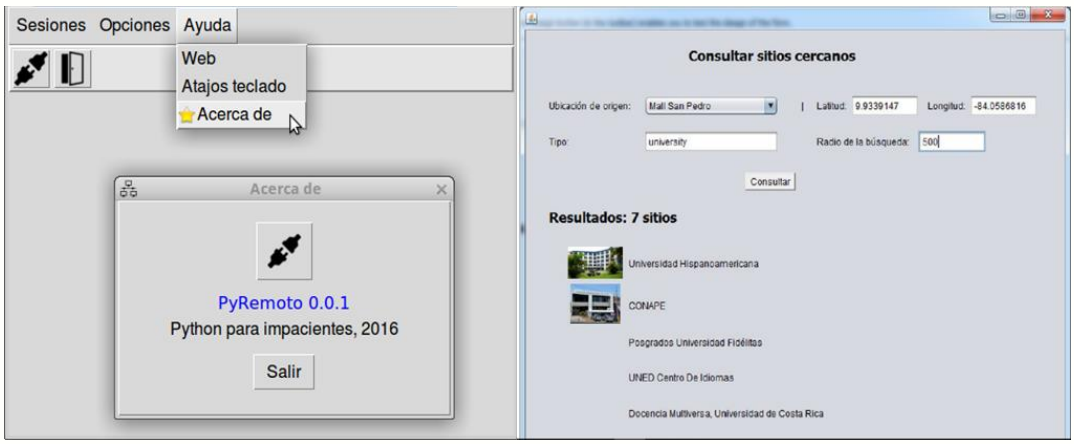
- Ejercicio de Herencia. Resolver en base al diagrama propuesto



1. La clase Persona debe tener los atributos nombre, apellido, ciudad, nacimiento.
2. La clase Profesor debe tener el atributo número de hora de clase semanal
3. La clase Alumno debe tener el atributo número de materias que está cursando.
4. La clase AlumnoBecado debe tener método que devuelva el valor de la matrícula ($150 por cada materia)
5. La clase AlumnoNoBecado debe tener método que devuelva el valor de la matrícula ($250 por cada materia)
6. La clase ProfesorTitular debe tener un método que devuelva su salario semanal (costo de hora clase $37.)
7. La clase ProfesorAuxiliar debe tener un método que devuelva su salario semanal (costo de hora clase $25.)
8. Se debe sobrescribir el método toString en las clases Alumno y Profesor, con características particulares de cada una de las clases.
9. Se debe crear una clase ejecutora que permita crear 4 objetos (AlumnoBecado, AlumnoNoBecado, ProfesorTitular, ProfesorAuxiliar), se deberá llamar al método toString y mostrar el valor de la matrícula o valor del salario semanal, según sea el caso.

- Realice una ventana en tkinter utilizando los Widget (con sus propiedades) más conocidos y que va a utilizar en adelante en el trascurso de la presente materia

Label, LabelFrame, Entry. Button, Combobox, Treeview, Scrollbar, Toplevel, Menu, Menubutton, Frame, Message, y los que considere necesario. Por ejemplo:



3.5. Autoevaluación 3

Responda a las siguientes preguntas según corresponda.

1. ¿Qué describen los métodos en una clase?
   a) La apariencia visual de los objetos de una clase.
   b) El comportamiento de los objetos de una clase.
   c) La estructura interna de los objetos de una clase.
   d) Las propiedades estáticas de los objetos de una clase.
2. ¿Qué es la sobrecarga de métodos?
   a) También conocida como Overriding Methods.
   b) Un proceso de asignación de nuevos valores a las variables.
   c) Una técnica para cambiar el nombre de un método.
   d) Un mecanismo para crear nuevos métodos en una clase.
3. ¿Qué hace la línea raiz.maonloop() en Tkinter?
   a) Crea la ventana principal.
   b) Inicia el bucle principal de eventos de la aplicación.
   c) Define los widgets en la ventana.
   d) Configura las propiedades de los widgets.
4. ¿Cuáles son las librerías utilizadas por el módulo Tkinter?
   a) TK, TTK, TTX
   b) Tk, ttk, tcl
   c) TK, TTK, Tcl
   d) Tk, Ttk, Tcl
5. ¿Cómo se posicionan los widgets en la ventana en Tkinter?
   a) Como si fuera una lista enlazada.

b) Usando coordenadas absolutas.

c) En una cuadrícula o matriz.

d) En un sistema de capas superpuestas.

6. ¿Qué es la herencia en programación orientada a objetos?

a) La capacidad de definir una nueva clase que es una versión modificada de otra ya existente.

b) La técnica para crear múltiples instancias de una clase.

c) El proceso de eliminar atributos de una clase base.

d) La habilidad de un objeto para cambiar de tipo dinámicamente.

7. ¿Cómo se llama a una función que no puede tomar parámetros con diferentes tipos?

a) Polimórfica

b) Sobrecargad.

c) Estática

d) Dinámica

8. ¿Cuál es la regla fundamental del polimorfismo?

a) Si todas las operaciones realizadas dentro de la función se pueden aplicar al tipo, la función se puede aplicar al tipo.

b) La función debe ser capaz de cambiar su firma dependiendo del tipo de parámetro.

c) Una función debe tener el mismo nombre, pero diferente número de parámetros.

d) Todas las funciones deben ser compatibles con los tipos de datos primitivos.

4. Unidad 4: El modelo de acceso a datos

Al instalar Python, se incluye automáticamente el controlador para conectarse a bases de datos creadas con SQLite. Sin embargo, si se requiere conectar a otros motores de bases de datos, es necesario instalar previamente el controlador correspondiente.

4.1. Conexión con base de datos

Por ello, el manejo de bases de datos en Python siempre sigue estos pasos:

1. Importar el conector
2. Conectarse a la base de datos (función connect del módulo conector)
3. Abrir un Cursor (método cursor de la conexión)
4. Ejecutar una consulta (método execute del cursor)
5. Obtener los datos (método fetch o iterar sobre el cursor)
6. Cerrar el cursor (método close del cursor)

4.2. Seguridad conexión – desconexión del sistema

La seguridad de una conexión a una base de datos depende en gran medida del equipo de desarrollo. En Python, existe una amplia variedad de librerías disponibles que permiten reforzar la seguridad de las conexiones.

4.2.1. Pasos para conectarse a una base de datos

Lo primero se debe hacer es importar el módulo del motor y luego crear un objeto de conexión para conectarnos a la base de datos. Este nos permitirá ejecutar las sentencias SQL.

En el ejemplo expuesto a continuación se utilizará SQLite, si se desea trabajar con otro motor se debe instalar la Librería.

- Un objeto de conexión se crea utilizando la función connect ():

```
import sqlite3
con = sqlite3.connect('mydatabase.db')
```

- Creación del cursor. - necesita un objeto cursor. Puedes crearlo utilizando el método cursor().

```
con = sqlite3.connect('mydatabase.db')
cursorObj = con.cursor()
```

- Si se desea crear tabla a la base de datos desde Python se lo puede realizar colocando sentencias SQL

```
def sql_table(con):
    cursorObj = con.cursor()
    cursorObj.execute("CREATE TABLE employees(id integer PRIMARY KEY, name text, salary real, department text, position text, hireDate text)")
    con.commit()
con = sql_connection()
sql_table(con)
```

- Para cerrar una conexión, utiliza el objeto de conexión y llame al método close () de la siguiente manera:

```
con = sqlite3.connect('mydatabase.db')
#program statements
con.close()
```

4.3. CRUD de datos

El término CRUD se refiere a las operaciones básicas en una base de datos: Crear (Create), Leer (Read), Actualizar (Update) y Eliminar (Delete).

- Crear datos:

```
import sqlite3
db=sqlite3.connect('test.db')
try:
    cur =db.cursor()
    cur.execute('''CREATE TABLE student (
    StudentID INTEGER PRIMARY KEY AUTOINCREMENT,
    name TEXT (20) NOT NULL,
    age INTEGER,
    marks REAL);''')
    print ('table created successfully')
except:
    print ('error in operation')
    db.rollback()
db.close()
```

- Leer datos

```
import sqlite3
db=sqlite3.connect('test.db')
sql="SELECT * from student;"
cur=db.cursor()
cur.execute(sql)
while True:
    record=cur.fetchone()
    if record==None:
        break
    print (record)
db.close()
```

- Actualizar datos

```
import sqlite3
db=sqlite3.connect('test.db')
qry="update student set age=? where name=?;"
try:
    cur=db.cursor()
    cur.execute(qry, (19,'Deepak'))
    db.commit()
    print("record updated successfully")
except:
    print("error in operation")
    db.rollback()
db.close()
```

- Eliminar datos

```
import sqlite3
db=sqlite3.connect('test.db')
qry="DELETE from student where name=?;"
try:
    cur=db.cursor()
    cur.execute(qry, ('Bill',))
    db.commit()
    print("record deleted successfully")
except:
    print("error in operation")
    db.rollback()
db.close()
```

Para más información visite la siguiente página web:
https://www.tutorialsteacher.com/python/database-crud-operation-in-python

4.4. Diseño de la arquitectura

Python es un lenguaje de programación interpretado, lo que le permite ejecutarse en cualquier sistema que tenga su intérprete instalado. Además de esta ventaja, Python ofrece diferentes dialectos, como Jython, que permite escribir código Python para ejecutarse en entornos Java.

4.4.1. Framework de gran utilidad

Python no solo es multiplataforma y multiparadigma, sino que también es adecuado para desarrollar todo tipo de aplicaciones, ya sean web o móviles. Para facilitar esto, Python cuenta con potentes Frameworks que abarcan desde el desarrollo web hasta la creación de juegos y algoritmos científicos para cálculos avanzados.

En un inicio Python fue diseñado para Unix, sin embargo, en la actualidad se puede correr con cualquier otro tipo de sistema. Cuando un lenguaje es multiparadigma permite la creación de no solo desarrollo web, si no también permite realizar aplicaciones o programas bajo otros criterios estructurales de código.

> Te recomiendo que des una lectura a la información oficial de Python en la siguiente página web:
> https://docs.python.org/3/

El Framework más conocida es Django, que sirve para el desarrollo de páginas web más rápido; existen muchos más Framework, pero en el trascurso de esta materia no se trabajará con los Framework, porque primero se debe aprender el lenguaje de Python y luego a los diferentes Framework que existen.

> Si está interesado en consultar y conocer el Framework Django visite la siguiente página web:
> https://docs.djangoproject.com/en/3.0/

4.5. Ejemplos y/o prácticas

Desarrollar en Python una base de datos que contenga las siguientes tablas que cuenta con los siguientes campos:

Países						
id	nombre	capital	Km	continente	lenguaje	n_habitantes
Alumno						
id	nombre	apellido	dirección	teléfono	celular	correo
Películas						
Id	nombre	descripción	Año	lenguaje	categoría	actor
Proveedor						
id	cedula	nombres	apellidos	dirección	teléfono	correo

Una vez realizado la base de datos desarrollar un programa en Python que permita hacer las siguientes operaciones:

- Insertar un registro dentro de una tabla
- Insertar un conjunto de registros dentro de una tabla (a menos 6 registros)
- Presentar los datos almacenados en la tabla
- Modificar a menos 2 registros de la tabla
- Eliminar al menos 2 registros
- Una vez ejecutadas las operaciones comentar lo que están desarrollando para que no les genere error
- Las variables que utilicen tienen que contener el nombre de ustedes

4.6. Autoevaluación 4

Responda a las siguientes preguntas según corresponda.

1. ¿Cuál es el propósito del método cursor() en Python?
   a) Para ejecutar una transacción.
   b) Para abrir una conexión a la base de datos.
   c) Para cerrar una conexión a la base de datos.
   d) Para crear una nueva base de datos.
2. ¿Qué significa el término CRUD en bases de datos?
   a) Create, Run, Update, Delete.
   b) Copy, Read, Update, Delete.
   c) Create, Read, Update, Delete.
   d) Compute, Read, Upload, Delete
3. ¿Qué realiza la sentencia SQL: INSERT INTO student (name, age, marks) VALUES('Rajeev', 20, 50);?

a) Elimina datos en la tabla.

b) Modifica datos existentes.

c) Crea nuevos datos en la tabla.

d) Selecciona datos de la tabla.

4. ¿Qué es SQLite?

a) Un motor de base de datos basado en servidor.

b) Un motor de base de datos relacional que no requiere configuración de servidor.

c) Una interfaz gráfica para gestionar bases de datos.

d) Una herramienta para administrar Oracle y MySQL.

5. ¿Para qué se utiliza SQLite?

a) Solo para bases de datos de servidor.

b) Exclusivamente como una base de datos integrada en dispositivos móviles, navegadores web y otras aplicaciones dependientes.

c) Para gestionar grandes bases de datos distribuidas.

d) Solo para sistemas de almacenamiento en la nube.

6. ¿Qué es Python Database API?

a) Un sistema de archivos integrado en Python.

b) Un conjunto de estándares recomendados para la estandarización de módulos de bases de datos.

c) Una librería de gráficos para bases de datos.

d) Una aplicación para crear interfaces gráficas.

7. ¿Qué hace el método commit() en una transacción?

a) Revierte la transacción al punto de partida.

b) Guarda de forma permanente los cambios realizados.

c) Cierra la conexión a la base de datos.

d) Crea una nueva transacción.

8. ¿Qué hace el método close() en una conexión de base de datos?

a) Revierte los cambios realizados en la transacción.

b) Cierra la conexión de forma permanente, generando un error si se intenta usar después.

c) Abre una nueva conexión de base de datos.

d) Ejecuta una transacción y guarda los cambios realizados.

5. Unidad 5: Informes

Los informes son esenciales en el mundo de la programación, ya que permiten llevar un control detallado de las acciones realizadas por un sistema de información. Python ofrece diversas librerías especializadas para generar reportes y documentos.

5.1. Reportes

ReportLab es un conjunto de herramientas de código abierto que permite la creación de documentos PDF utilizando Python. Se trata de una librería muy extensa que tiene muchas aplicaciones, desde pequeños textos y figuras geométricas hasta grandes gráficos e ilustraciones que se pueden descargar en un PDF.

La librería se instala sencillamente vía pip:

```
pip install reportlab
```

Ahora da lectura a la página web:
https://recursospython.com/guias-y-manuales/crear-documentos-pdf-en-python-con-reportlab/ https://docs.reportlab.com/

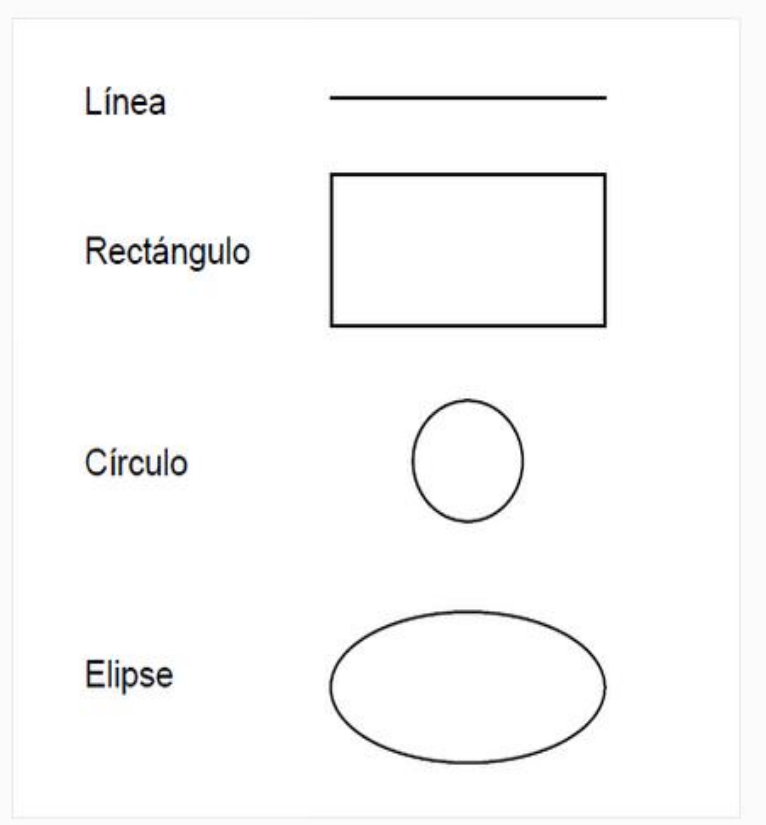
ReportLab ofrece un lenguaje de plantillas de alto nivel llamado RML, que se asemeja a HTML y a los sistemas de plantillas utilizados en el desarrollo web, y una API de bajo nivel para generar documentos PDF directamente desde Python. Para aquellos que deben utilizar ampliamente las habilidades de la librería al crear documentos, la segunda opción generalmente resulta más conveniente. El código más básico que podremos encontrar usando ReportLab es aquel que genera un documento PDF vacío, como por ejemplo el siguiente código que se muestra en pantalla.

```
1. from reportlab.pdfgen import canvas
2.
3. c = canvas.Canvas("hola-mundo.pdf")
4. c.save()
```

5.1.1. Líneas geométricas

ReportLab facilita el dibujo de líneas, rectángulos, círculos y otras figuras. Al combinar estas herramientas, podemos generar un documento PDF como el siguiente.

```python
from reportlab.lib.pagesizes import A4
from reportlab.pdfgen import canvas

w, h = A4
c = canvas.Canvas("figuras.pdf", pagesize=A4)
c.drawString(30, h - 50, "Línea")
x = 120
y = h - 45
c.line(x, y, x + 100, y)
c.drawString(30, h - 100, "Rectángulo")
c.rect(x, h - 120, 100, 50)
c.drawString(30, h - 170, "Círculo")
c.circle(170, h - 165, 20)
c.drawString(30, h - 240, "Elipse")
c.ellipse(x, y - 170, x + 100, y - 220)
c.showPage()
c.save()
```



5.1.2. Grilla

El método grid() de ReportLab nos facilita el trabajo al generar grillas en lugar de usar line() o lines(), que toma como primer argumento una lista de posiciones en X y como segundo argumento una lista de posiciones en Y.

```python
import itertools
from random import randint
from statistics import mean

from reportlab.lib.pagesizes import A4
from reportlab.pdfgen import canvas


def grouper(iterable, n):
    args = [iter(iterable)] * n
    return itertools.zip_longest(*args)


def export_to_pdf(data):
    c = canvas.Canvas("grilla-alumnos.pdf", pagesize=A4)
    w, h = A4
    max_rows_per_page = 45
    # Margin.
    x_offset = 50
    y_offset = 50
    # Space between rows.
    padding = 15

    xlist = [x + x_offset for x in [0, 200, 250, 300, 350, 400, 480]]
    ylist = [h - y_offset - i*padding for i in range(max_rows_per_page + 1)]

    for rows in grouper(data, max_rows_per_page):
        rows = tuple(filter(bool, rows))
        c.grid(xlist, ylist[:len(rows) + 1])
        for y, row in zip(ylist[:-1], rows):
            for x, cell in zip(xlist, row):
                c.drawString(x + 2, y - padding + 3, str(cell))
        c.showPage()

    c.save()


data = [("NOMBRE", "NOTA 1", "NOTA 2", "NOTA 3", "PROM.", "ESTADO")]
for i in range(1, 101):
    exams = [randint(0, 10) for _ in range(3)]
    avg = round(mean(exams), 2)
    state = "Aprobado" if avg >= 4 else "Desaprobado"
    data.append((f"Alumno {i}", *exams, avg, state))
export_to_pdf(data)
```

Alumno 90	10	4	8	7.33	Aprobado
Alumno 91	10	3	6	6.33	Aprobado
Alumno 92	4	7	10	7	Aprobado
Alumno 93	6	10	7	7.67	Aprobado
Alumno 94	10	7	9	8.67	Aprobado
Alumno 95	2	4	8	4.67	Aprobado
Alumno 96	5	8	0	4.33	Aprobado
Alumno 97	9	3	7	6.33	Aprobado
Alumno 98	9	4	9	7.33	Aprobado
Alumno 99	1	4	0	1.67	Desaprobado
Alumno 100	8	9	3	6.67	Aprobado

Como se mencionó al principio de la unidad, hemos examinado las funciones principales de ReportLab, pero solo una pequeña parte de su amplia gama de características. Aquellos que necesiten utilizar la librería de manera más intensiva ya conocerán las bases y serán dirigidos a la documentación oficial para familiarizarse con las herramientas más complejas.

5.2. Ejemplos y/o prácticas

Realice la siguiente ventana utilizando Tkinter y luego presentar su respectivo reporte de acuerdo con los datos ingresados en la ventana.

5.3. Autoevaluación 5

Responda a las siguientes preguntas según corresponda.

1. ¿Qué es ReportLab?

   a) Un software comercial para crear documentos PDF desde Python.
   b) Un toolkit de código abierto para crear documentos PDF desde Python.
   c) Una librería propietaria para edición de imágenes en Python.
   d) Un entorno de desarrollo integrado (IDE) para Python.

2. ¿Qué características ofrece la librería ReportLab para generar documentos PDF?

   a) Una API de alto nivel para crear documentos PDF y un lenguaje de plantillas llamado RML.
   b) Una API de bajo nivel para crear documentos PDF y un lenguaje de plantillas similar a HTML llamado RML.
   c) Solo un lenguaje de plantillas de alto nivel llamado RML.

d) Una API de alto nivel para generar documentos HTML y un lenguaje de plantillas similar a PDF.

3. ¿Cómo se comparan los productos comerciales y de código abierto de ReportLab con otros sistemas de documentación?

a) Son completamente compatibles con una amplia gama de documentación.
b) Son incompatibles con una amplia gama de documentación.
c) Solo son compatibles con documentación en formato PDF.
d) Son parcialmente compatibles con documentación en formato HTML.

4. ¿Bajo qué licencia está disponible la versión de código abierto del kit de herramientas ReportLab y otras herramientas de código abierto?

a) Licencia MIT.
b) Licencia GPL.
c) Licencia BSD.
d) Licencia Apache.

5. ¿Qué hace el método setStrokeColorRGB() en ReportLab?

a) Establece el color de fondo de las figuras.
b) Establece el color del borde de las figuras.
c) Cambia el color del texto.
d) Ajusta el grosor del borde de las figuras.

6. ¿Cómo se mantiene la fuente y el tamaño del texto al usar drawString() en ReportLab?

a) Mediante el método setFont().
b) Mediante el método setTextStyle().
c) Mediante el método setTextColor().
d) Mediante el método **setTextSize().**

7. ¿Qué método proporciona ReportLab para facilitar la generación de grillas?

a) createGrid()
b) drawGrid()
c) grid()
d) generateGrid()

8. ¿Qué librería utiliza ReportLab para insertar imágenes en documentos PDF?

a) NumPy
b) Matplotlib

c) Pillow

d) OpenCV

Solucionarios

Autoevaluación unidad 1	
1.	B
2.	A
3.	C
4.	B
5.	B
6.	A

Autoevaluación unidad 2	
1.	B
2.	A
3.	C
4.	A
5.	A
6.	A
7.	A
8.	A
9.	A

Autoevaluación unidad 3	
1.	B
2.	A
3.	B
4.	C
5.	C
6.	A
7.	A
8.	A

Autoevaluación unidad 4	
1.	B
2.	C
3.	C
4.	B
5.	B
6.	B
7.	B
8.	B

Autoevaluación unidad 5	
1.	B
2.	B
3.	B
4.	C
5.	B
6.	A
7.	C
8.	C

Referencias

- CACES. (2021). MODELO DE EVALUACIÓN EXTERNA 2024 CON FINES DE ACREDITACIÓN PARA LOS INSTITUTOS SUPERIORES TÉCNICOS Y TECNOLÓGICOS. Quito: Consejo de Aseguramiento de la Calidad de la Educación Superior.

Diaz, D. (2023). Kinsta. Ontenido de 25 frameworks de python para dominar. https://kinsta.com/es/blog/python-frameworks/

Huet Pablo. (2024). Fundamentos de python: Sintaxis, variables y estructuras | OpenWebinars. https://openwebinars.net/blog/fundamentos-de-python-sintaxis-variables-y-estructuras-de-control/

Philipp Acsany. (2024). Real Python. Obtenido de What Are CRUD Operations?: https://realpython.com/crud-operations/

- Real Academia Española. (s. f.). expresión. Diccionario de la lengua española. Obtenido 17 de septiembre de 2024, de https://dle.rae.es/expresi%C3%B3n

Vasallo, R. (2021). Python: El lenguaje de programación más versátil y demandado. Revista de Tecnología y Programación, 12(3), 45-56.

SOLUZIONINNOVATIVE S.A.S.

EDITORIAL

editorialsoluzioniinnovative@gmail.com
https://soluzioninnovativegroup.com/repositorio/

Juan Pablo Pardo

Destacado profesional en el ámbito de la informática y la educación, con una trayectoria académica y experiencia en la enseñanza y desarrollo de tecnologías educativas. Ingeniero en informática por la Universidad Técnica Particular de Loja y Máster en Tecnologías Educativas y Competencias Digitales por la Universidad Internacional de La Rioja. Docente investigador de la especialidad de Desarrollo de Software del Instituto Superior Tecnológico Mariano Samaniego; ha participado en cursos y talleres; posee varias publicaciones en revista científicas relacionadas con las metodologías activas en la educación; así mismo, ha participado en varios congresos de divulgación científica, además ha desarrollado aplicantes web en beneficio del establecimiento donde presta sus servicios.

ISBN: 978-9942-7294-2-2