

FULL STACK EN 16 SEMANAS

Antes de empezar...

Esta Guía trata de reforzar y complementar semana a semana las clases en vivo que reciben los estudiantes del Bootcamp de codiGo-Tecsup que aprenden con dedicación, esfuerzo y sobre todo mucha practica lo que los mentores de codiGo. expertos en la industria los capacitan con proyectos reales y sobre todo mucha dedicación. no es necesario saber de sistemas o haber hecho un programa por computadora para aprender con nosotros, porque dentro de nuestra primera promoción hemos tenido estudiantes de distintas carreras: Chef, Ingenieros Civiles, Ingenieros Pesqueros, Arquitectos, Desarrollador de Videos juegos etc. pero todos ellos tuvieron algo en común... la pasión por la tecnología y ser parte de ella misma.

Empezemos.....



Python
Instalación
Sintaxis
Semántica

MicroFrameWork
Flask

Python
Accediendo a
Base de datos

Framework
DJANGO

CODIGO
Santa Anita, Avenida
Cascañueces 2221,
Lima 15011

www.codigo.edu.pe

21/09/2019



INDICE

3. Palabras claves
4. Primeros Pasos con Python
5. Sintaxis y Semántica
6. Sintaxis y Semántica
7. Sintaxis y Semántica
8. Sintaxis y Semántica
9. Ejercicios Propuestos
10. Flask
11. Flask
12. Flask
13. Django
14. Django
15. Django
16. Django
17. Nodejs
18. Nodejs
19. Nodejs
20. Firebase
21. Firebase
22. Microservicios
23. Microservicios
24. Microservicios
25. DRF
26. DRF
27. DRF
28. MYSQL
29. MYSQL
30. MONGODB
31. MONGODB
32. MONGODB
33. CLOUD
34. CLOUD
35. JWT
36. PASSPORT



ANTES DE EMPEZAR



HOLA ME LLAMO GUILIANO Y SOY UN EX -ALUMNO DE CODIGO, SERE TU AVATAR QUE TE AYUDARA A GUIARTE PASO A PASO EN EL RECCORIDO DE ESTA GUIA, ESPERO SERVIRTE DE MUCHO ES MI MISION, ¡SI NO DANNY YA NO ME VA A CONTRATAR!

Pero Antes de empezar te invito revisar las siguientes palabras reservadas que te servirán en el contenido de esta guía

PALABRAS CLAVES

- Bootcamp.** - Son una nueva modalidad formativa que no exige un título académico previo, sino ganas de aprender y unos mínimos conocimientos, son muy intensivos y focalizado a la práctica permiten insertarte a un mercado laboral en poco tiempo.
- Mentor.** - Es una persona más experimentada o con mayor conocimiento que ayuda a otro menos experimentada a manera de practica y dedicación
- Backend.** -Es la parte del desarrollo web que se encarga de que toda la lógica de una página web funcione. Se trata de un conjunto de acciones que pasan en una web pero que no vemos como por ejemplo la ejecución de microservicios
- FrontEnd.** - Es la parte del desarrollo web que interactúa con los usuarios y el Backend, procesa la entrada y tiene que ver con la buena experiencia de usuario, se encarga del diseño y la funcionalidad de la misma.
- FullStack.** - Es un programador que conoce muy bien tanto la parte del FrontEnd y el Backend, se maneja en sistemas y sabe entender es autodidacta.
- Autodidacta.** -Es la persona que se instruye y aprende nuevos conocimientos a través de sus propios medios como libros, internet o foros.
- Lenguaje de Programación.** -Es un Lenguaje Formal que proporciona una serie de instrucciones que permiten aun programador escribir secuencias de órdenes y algoritmos a modo de controlar el comportamiento físico y lógico de una computadora con el objetivo que produzca diversas clases de datos.
- Código.** -Es un conjunto de líneas de texto con los pasos que debe seguir la computadora para ejecutar un programa.
- Compilador.** -Es un tipo de traductor que transforma un programa entero (Código Fuente) a código de máquina, para que el computador lo pueda ejecutar.
- Algoritmo.** -Es un conjunto de instrucciones o reglas definidas y no ambiguas, ordenadas y finitas que permiten solucionar un problema.
- Sintaxis.** -La sintaxis de un lenguaje de programación se define como el conjunto de reglas que se deben seguirse al escribir el código de los programas, para considerarse como correctos para dicho lenguaje de programación es la gramática propia.
- Semántica.** - Se refiere al significado que tiene los elementos de un lenguaje de programación.
- Cloud.** - Es un paradigma que te permite ofrecer servicios de computación a través de una red que usualmente es internet



¡¡¡Empezamos
con la semana 1
en ella
veremos!!!

SEMANA 1



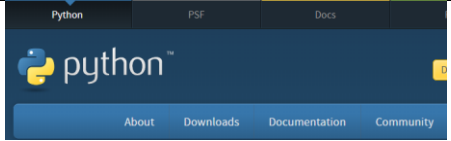
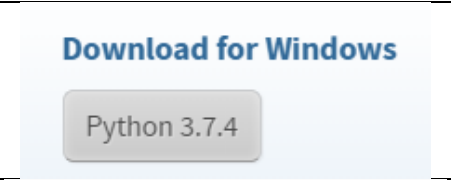
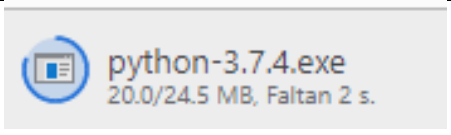


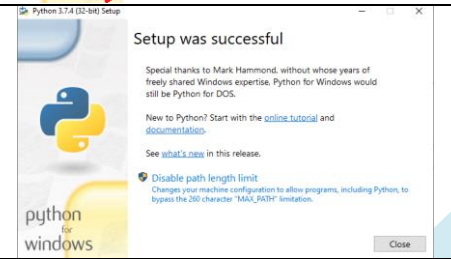
1. PRIMEROS PASOS CON PYTHON:

- A. INSTALACION
- B. SINTAXIS Y SEMANTICA

Primeros Pasos con Python

Python es un lenguaje de programación con las siguientes características: Interpretado, Multiparadigma, Multiplataforma, Multipropósito, OpenSource que nos permite escribir programas para todo propósito. para poder trabajar con el deberemos descargar el interprete y comenzar con su instalación, para esto deberemos tener una computadora con conexión a Internet y ingresar a nuestro navegador preferido. Comencemos con la instalación.

A. Instalación de Python

1. Ingresar a la página www.python.org desde una computadora	
2. Ingresar al Menú Downloads (Descargar) y presionar clic en el botón Python 3.7.4	
3. Se procederá a descargar el interprete de Python en la parte inferior izquierda de su pantalla	
4. Una Vez descargado el intérprete, dirjase a la carpeta descargas de su computador y pulse doble clic en el icono que se muestra en la imagen	
5. Luego deberá pulsar la opción Add "Python 3.7 to PATH" para agregar Python a sus variables de entorno a continuación "Install Now"	
6. Una vez terminado la instalación deberá mostrarle la siguiente pantalla	


"Felicitaciones Ud. acaba de instalar el interprete de Python"



¡¡¡Vamos
Aprender ahora
la sintaxis y
semántica a
Codear!!!

B. Sintaxis y Semántica

Ahora aprendernos a escribir código en Python para eso deberemos aprender su sintaxis y semántica, sigamos con los siguientes pasos:

1. Nos vamos al menú de inicio de nuestro computador y buscamos la carpeta Python 3.7 en ella encontraremos en IDLE de Python presiona clic en el	
2. Nos cargamos el IDLE de Python en el escribiremos nuestro primer Hola Mundo, luego de escribir pulsa enter.	<pre>Python 3.7.4 (tags/v3.7.4:09359112e, Jul 8 2019, 19:29:22) [MSC v.191 Type "help", "copyright", "credits" or "license()" for more information. >>> print("Hola Mundo desde CodiGO") Hola Mundo desde CodiGO >>> </pre>
“Felicitaciones Ud. acaba de Escribir su primer programa en Python”	



“Ahora Con la ayuda del IDLE de Python comenzaremos a revisar su sintaxis y semántica a más detalle”:

```
>>> #Esto es un comentario y nos ayudara a describir lo que
estamos codeando
>>> # 1 Asignación de Variables
>>> mentor = "Sebastian"
>>> edad = 35
>>> sueldo = 7800.20
>>> casado = True
>>> # 2 La Función type no devuelve el tipo de datos que
contiene una variable
>>> type(mentor)
<class 'str'>
>>> type(edad)
<class 'int'>
>>> type(sueldo)
<class 'float'>
>>> type(casado)
<class 'bool'>
```

>>> # 3 Operadores Matemáticos

```
>>> suma = 20 + 89
>>> suma
109
>>> resta = 10 - 8
>>> resta
2
>>> multi = 34 * 8
>>> multi
272
>>> divdec = 24 / 7
>>> divdec
3.4285714285714284
>>> divent = 24 // 7
>>> divent
3
>>> res = 24 % 7
>>> res
3
>>>
```



>>> # 4 Operadores Lógicos

```
>>> r = 5 == 3
>>> r
False
>>> r != 3
True
>>> r < 3
True
>>> r > 3
False
>>> r <= 3
True
>>> r >= 3
False
>>>
```

>>> # 5 Conversión de tipo de datos

```
>>> str(23)
'23'
>>> str(True)
'True'
>>> float(12)
12.0
>>> int('123')
123
>>> bool(0)
False
>>> bool(-5)
True
>>> bool(5)
True
>>>
```

>>> # 6 Constantes

```
>>> igv = 0.18
>>> des = 10
>>> precio = 200.00
>>> cant = 5
>>> subtotal = precio * cant
>>> subtotal
1000.0
>>> vigv = subtotal * igv
>>> vigv
180.0
>>> total = (subtotal + vigv) - des
>>> total
1170.0
>>>
```

>>> # 7 Formato de Cadena

```
>>> nombre = "Armando"
>>> print("Hola, %s!" % nombre)
Hola, Armando!
>>> edad = 39
>>> print("%s tiene %d años." % (nombre, edad))
Armando tiene 39 años.
>>> tipo_calculo = "raíz cuadrada de dos"
>>> valor = 2**0.5
>>> print("el resultado de %s es %.8f" % (tipo_calculo,
valor))
el resultado de raíz cuadrada de dos es 1.41421356
>>>
```



Python conoce varios tipos de datos compuestos, que se usan para agrupar otros valores. La más versátil es la lista, que puede escribirse como una lista de valores separados por comas (elementos) entre corchetes. Las listas pueden contener elementos de diferentes tipos, pero generalmente todos los elementos tienen el mismo tipo

>>> # 8 Listas

```
>>> miseleccion = ["GALLESE", "TAPIA", "YOTUN"]
>>> miseleccion
['GALLESE', 'TAPIA', 'YOTUN']
>>> miseleccion.append("FLORES")
>>> miseleccion
['GALLESE', 'TAPIA', 'YOTUN', 'FLORES']
>>> miseleccion.insert(2, "AQUINO")
>>> miseleccion
['GALLESE', 'TAPIA', 'AQUINO', 'YOTUN', 'FLORES']
>>> miseleccion.pop()
'FLORES'
>>> miseleccion
['GALLESE', 'TAPIA', 'AQUINO', 'YOTUN']
>>> miseleccion.pop(2)
'AQUINO'
>>> miseleccion
['GALLESE', 'TAPIA', 'YOTUN']
>>> miseleccion[1]
'TAPIA'
>>> miseleccion[1] = "TRAUCO"
>>> miseleccion
['GALLESE', 'TRAUCO', 'YOTUN']
>>> miseleccion.remove("TRAUCO")
>>> miseleccion
['GALLESE', 'YOTUN']
>>> miseleccion.sort()
>>> miseleccion
['GALLESE', 'YOTUN']
>>> miseleccion.reverse()
>>> miseleccion
['YOTUN', 'GALLESE']
>>> miseleccion.count("YOTUN")
1
>>>
```



Una sentencia condicional es una instrucción o grupo de instrucciones que se pueden ejecutar o no en función del valor de una condición

>>> # 9 Condicionales

```
>>> numero = 1
```

```
>>> if numero > 0 :
```

```
    print("El numero es positivo")
```

pulsa enter nuevamente acá para que se genere el resultado

```
El numero es positivo
```

```
>>> numero = -1
```

```
>>> if numero>0:
```

```
    print("El Numero es positivo")
```

```
else:
```

```
    print("El Numero es negativo")
```

pulsa enter nuevamente acá para que se genere el resultado

```
El Numero es negativo
```

```
>>>
```

```
>>> area = "SISTEMAS"
```

```
>>> if area == "MARKETING":
```

```
    print("AREA INCORRECTA")
```

```
elif area == "RRHH":
```

```
    print("AREA INCORRECTA")
```

```
elif area == "SISTEMAS":
```

```
    print("AREA VALIDA")
```

pulsa enter nuevamente acá para que se genere el resultado

```
AREA VALIDA
```

```
>>>
```

>>> # 9.1 Condicionales

```
>>> x = int(input("Ingrese un numero entero: "))
```

```
Ingrese un numero entero: 2
```

```
>>> if x < 0:
```

```
    x = 0
```

```
    print("Numero Negativo cambiado a cero")
```

```
elif x== 0:
```

```
    print("Cero")
```

```
elif x== 1:
```

```
    print("simple")
```

```
else:
```

```
    print("Mayor de un digito")
```

```
Mayor de un digito
```



Un bucle o ciclo, en programación, es una secuencia que ejecuta repetidas veces un trozo de código, hasta que la condición asignada a dicho bucle deja de cumplirse. Los dos bucles más utilizados en Python

>>> # 10 Procesos Repetitivos

```
>>> edad =0 #bucles while
```

```
>>> while edad <18:
```

```
    edad = edad +1
```

```
    if (edad %2) == 0:
```

```
        continue
```

```
    print("Felicitaciones tienes" + str(edad))
```

```
Felicitaciones tienes1
```

```
Felicitaciones tienes3
```

```
Felicitaciones tienes5
```

```
Felicitaciones tienes7
```

```
Felicitaciones tienes9
```

```
Felicitaciones tienes11
```

```
Felicitaciones tienes13
```

```
Felicitaciones tienes15
```

```
Felicitaciones tienes17
```

```
>>>
```

>>> # 10.1 Procesos Repetitivos

```
>>> for i in range (1,20):
```

```
    print ("Hola Soy un bucle")
```

```
Hola Soy un bucle
```

```
Hola Soy un bucle
```

```
Hola Soy un bucle
```

```
Hola Soy un bucle
```

```
Hola Soy un bucle
```

```
Hola Soy un bucle
```

```
Hola Soy un bucle
```

```
Hola Soy un bucle
```

>>> # 10.2 Procesos Repetitivos

```
>>> for i in range (1,20):
```

```
    if i == 10:
```

```
        break
```

```
    else:
```

```
        print("No Llego a Diez aun")
```

```
    continue
```

>>> # 10.3 Procesos Repetitivos

```
>>> miseleccion =["GALLESE","TAPIA","YOTUN"]
```

```
>>> for i in range (len(miseleccion)):
```

```
    print(i,miseleccion[i])
```

```
0 GALLESE
```

```
1 TAPIA
```

```
2 YOTUN
```

```
>>>
```



Las Palabras reservadas son la que trae el propio interprete como propias que el usuario no podrá utilizar como parte de su código

```
>>> # 10.4 Palabras Reservadas
>>> import keyword
>>> keyword.kwlist
['False', 'None', 'True', 'and', 'as', 'assert', 'async', 'await',
'break', 'class', 'continue', 'def', 'del', 'elif', 'else', 'except',
'finally', 'for', 'from', 'global', 'if', 'import', 'in', 'is',
'lambda', 'nonlocal', 'not', 'or', 'pass', 'raise', 'return',
'try', 'while', 'with', 'yield']
>>> keyword.iskeyword('While')
False
>>> keyword.iskeyword('while')
True
>>>
```



Las funciones son piezas de código delimitadas y a las que se les puede asignar un nombre con el que pueden ser invocadas. Las funciones son uno de varios tipos invocables (callable) de Python.

```
>>> # 10.5 Funciones
>>> def sumar():
>>>     print (5+10)
>>> sumar()
15

>>> def sumar (valor1, valor2):
>>>     print (valor1+valor2)
>>> sumar(10,15)
25

>>> def sumarlistas(numeros):
>>>     res = 0
>>>     for n in numeros:
>>>         res +=n
>>>     print (res)

>>> sumarlistas([23,45,4])
72
>>>
```



Python posee una serie de librería que podemos incluir en nuestro código a continuación les mostrare algunas si desea más información no dude consultar la documentación oficial:
<https://docs.python.org/3/library/index.html>

```
>>> # 10.6 Librería Estándar de Python
>>> import math
>>> math.pi
3.141592653589793

>>> math.cos(math.pi / 4)
0.7071067811865476

>>> import random
>>> random.choice(['cueva','yotun','advincula'])
'advincula'

>>> random.random()
0.5488025286121495

>>> random.randrange(5)
1

>>> import statistics
>>> datos = [2.75,1.75,1.25,0.25,0.5,1.25,3.5]
>>> statistics.mean(datos)
1.6071428571428572

>>> statistics.median(datos)
1.25

>>> statistics.variance(datos)
1.3720238095238095

>>> from datetime import date
>>> hoy = date.today()
>>> hoy

datetime.date(2019, 9, 20)
>>> nacimiento = date(1964,7,31)
>>> edad = hoy - nacimiento
>>> edad.days
20139
```



¡¡¡Felicitaciones ya terminamos la primera semana que te pareció, fácil ¡no!, ahora nos queda repasar creando ejercicios manos a la Obra¡!!



ARTILLERIA DE EJERCICIOS SEMANA 1

1. Crear una Función que retorne el mayor de dos números o 0 si son iguales
2. Crear una Función que determine si una persona es mayor de edad o no (pista: el retorno debe ser un valor booleano)
3. Programar una función que determine si una empresa es microempresa o no (retorno booleano –True o False–). Se dice que es microempresa si tiene menos de 50 empleados, factura menos de 30 millones de soles y tiene un balance igual o inferior a los 5 millones de soles.
4. Escribamos un programa para solicitar al usuario el número de horas y el precio por hora con vistas a calcular su salario bruto. Las horas que sobrepasen 40 se considerarán extra y pagadas a 1,5 veces el precio de la hora regular.
5. Escriba una función que dada una nota expresada en forma numérica devuelva uno de los siguientes valores, definidos en un enumerado: SUSPENSO, APROBADO, NOTABLE, SOBRESALIENTE, VALOR_INCORRECTO. La correspondencia de los valores numéricos es la siguiente:
 $0 \leq \text{nota} < 5 = \text{SUSPENSO}$; $5 \leq \text{nota} < 7 = \text{APROBADO}$;
 $7 \leq \text{nota} < 9 = \text{NOTABLE}$; y $9 \leq \text{nota} \leq 10 = \text{SOBRESALIENTE}$. Cualquier otro valor numérico se corresponderá con VALOR INCORRECTO.
6. Crear un programa que permite que el usuario deberá ingresar el número de horas trabajadas y el valor por cada hora. Considere en los cálculos el descuento de seguridad social 9,35% sobre el total de ingresos y una bonificación del 2% del sueldo inicial a percibir.
7. Realice un programa que obtenga el índice de masa corporal de una persona, ingresando la estatura en centímetros y el peso en kilos.
8. Crear un programa que contiene una Lista de 30 notas de 0 a 20 generados aleatoriamente y mostradas en pantalla, de acuerdo a la nota contenida, indique cuántos estudiantes son:
Deficientes 0-5
Regulares 6-10
Buenos 11-15
Excelentes 16-20
9. Crea un programa que pida al usuario 10 palabras, las guarde en una lista y luego le pregunte de forma repetitiva qué texto quiere buscar. Le responderá si dicho texto era alguno de lo que se habían introducido inicialmente. Dejará de repetirse cuando se introduzca "fin".
10. Crear un programa que Sea capaz de almacenar datos de hasta 10 personas en una lista (sólo el nombre y el número de teléfono). Aparecerá un menú en pantalla que permita añadir una nueva persona, ver los nombres de todas las personas almacenadas, ver el teléfono de una cierta persona
11. (a partir de su nombre), corregir los datos de una persona, eliminar los datos de una persona, salir.
12. Un comerciante tiene 4 tiendas en diferentes partes de la capital: Sta. Anita, Chorrillos, Ate, Lince y Surco deberá ingresar en una lista las ventas generadas mensualmente por todo el 2018 por cada tienda, crear un menú de opciones donde me indique los Siguiente:
 - a. El Media de ventas de todas las tiendas
 - b. La Tienda que genero más en el mes de febrero
 - c. La Tienda que genero más en el mes de diciembre y Julio
13. Sebastián tiene una gran colección de música, y es muy particular con las canciones que escucha. Cada canción tiene un nombre, que es una cadena de caracteres. Su reproductor de música NO tiene una función de búsqueda que permite a Sebastián escribir una subcadena en el cuadro de búsqueda, y el reproductor, a continuación, enumera todas las canciones cuyos nombres contengan la subcadena. Si no es exactamente una canción que coincide con la búsqueda, entonces Sebastián lo puede pulsar la tecla Intro para escuchar esa canción.
Sebastián odia usar el ratón, y que no le gusta escribir mucho, así que insiste en escribir siempre la subcadena más breve posible que coincide exactamente con la canción que quiere escuchar en este momento. ¿Podrías ayudarlo a crear esa función de búsqueda óptima?
14. Usted es empresario en Cajamarca, y tiene la brillante idea de abrir una tienda de la leche en la Plaza Mayor. Como es una persona muy prudente, desea que la leche que venderá sea perfectamente natural y fresca, y por esa razón, va a traer unas sanísimas vacas de la región de Chota a Celendín. Tiene a su disposición un camión con un cierto límite de peso, y un grupo de vacas disponibles para la venta. Cada vaca puede tener un peso distinto, y producir una cantidad diferente de leche al día.
Su objetivo como empresario es elegir qué vacas comprar y llevar en su camión, de modo que pueda maximizar la producción de leche, observando el límite de peso del camión.
Entrada: Número total de vacas en la zona de Chota que están a la venta.
Entrada: Peso total que el camión puede llevar.
Entrada: Lista de pesos de las vacas.
Entrada: Lista de la producción de leche por vaca, en litros por día.
Salida: Cantidad máxima de producción de leche se puede obtener.



La P. OO es un paradigma de programación que viene a innovar la forma de obtener resultados. Los objetos manipulan los datos de entrada para la obtención de datos de salida específicos, donde cada objeto ofrece una funcionalidad

```
class MiPrimeraClase:
    pass
```

```
>>> a = MiPrimeraClase()
>>> b = MiPrimeraClase()
>>> print(a)
<__main__.MiPrimeraClase object at 0x038E6170>
>>> print(b)
<__main__.MiPrimeraClase object at 0x03DE2450>
```

```
class Cursos:
```

```
    #definimos la característica del objeto
    def __init__(self,nombre,precio,dura):
        self.nombre = nombre
        self.precio =precio
        self.dura = dura
    def imprimirdetalle(self):
        print(self.nombre,self.precio,self.dura)
```

```
>>> curso1= Cursos("python",200,"1mes")
>>> curso2 = Cursos("django",300,"2meses")
>>> curso3 = Cursos("Flask",800,"1mes")
>>> curso1.imprimirdetalle()
python 200 1mes
>>> curso2.imprimirdetalle()
django 300 2meses
>>> curso3.imprimirdetalle()
Flask 800 1mes
```

```
>>> class Cuenta:
```

```
    def __init__(self, valor):
        self.cantidad = valor
    def depositar(self, valor):
        if valor > 0:
            self.cantidad = self.cantidad + valor
        else:
            print ("El valor para depositar es
erroneo::")
    def mostrarDetalles(self):
        print ("La cantidad de la cuenta:",
self.cantidad)
```

```
>>> class Cliente :
```

```
    def __init__( self, n, d, e, c ) :
        self.nombre = n
        self.direccion = d
        self.edad = e
        self.cuenta = c
    def mostrarDetalles( self ) :
```

```
print ( "Nombre::", self.nombre )
print ( "Direccion::", self.direccion )
print ( "Edad::", self.edad )
self.cuenta.mostrarDetalles()
```

```
>>> cuenta1 = Cuenta( 300 )
>>> cuenta1.mostrarDetalles()
La cantidad de la cuenta: 300
>>> cuenta1.depositar( 400 )
>>> cuenta1.mostrarDetalles()
La cantidad de la cuenta: 700
>>> cliente = Cliente( "Virginia", "direccion", 25, cuenta1)
>>> cliente.mostrarDetalles()
Nombre:: Virginia
Direccion:: direccion
Edad:: 25
La cantidad de la cuenta:: 700
```

Este ejemplo anterior muestra las relaciones entre clases, del tipo "tiene un (a)"
El Cliente tiene una Cuenta.

1. Se tiene que poner un atributo del Cuenta en la clase Cliente
2. En los métodos del Cliente se tienen que usar los métodos de la Cuenta
3. En las pruebas se hace referencia de un objeto Cuenta al objeto Cliente.

```
>>> class Persona:
```

```
    def __init__(self,edad,nombre):
        self.edad = edad
        self.nombre = nombre
        print("Se ha Creado a ",self.nombre," de ",
self.edad," años.")
```

```
    def hablar(self, *palabras):
        for frase in palabras:
            print(self.nombre," : ",frase)
```

```
>>> # usaremos herencia para crear la clase deportista
```

```
>>> class Deportista(Persona):
    def practicaDeporte(self):
        print(self.nombre," : voy a practicar ")
```

```
>>> armando = Persona(37,"Armando")
Se ha Creado a Armando de 37 años.
>>> armando.hablar("Hola, estoy hablando ","Este Soy yo")
>>> sebas = Deportista(24,"Sebastian")
Se ha Creado a Sebastian de 24 años.
>>> sebas.practicaDeporte()
Sebastian : voy a practicar
>>>
```



```
>>> class Vehiculo():
    def __init__(self, color, ruedas):
        self.color = color
        self.ruedas = ruedas
    def __str__(self):
        return "color {}, {}
ruedas".format( self.color, self.ruedas )

>>> class Coche(Vehiculo):
    def __init__(self, color, ruedas, velocidad,
cilindrada):
        self.color = color
        self.ruedas = ruedas
        self.velocidad = velocidad
        self.cilindrada = cilindrada
    def __str__(self):
        return "color {}, {} km/h, {} ruedas, {}
cc".format( self.color, self.velocidad, self.ruedas,
self.cilindrada )

>>> c = Coche("azul", 150, 4, 1200)
>>> c
<__main__.Coche object at 0x0426FD10>
>>> print(c)
color azul, 4 km/h, 150 ruedas, 1200 cc
>>>

>>> class Ejemplo:
    __atributo_privado = "Soy un atributo inalcanzable
desde fuera"
    def __metodo_privado(self):
        print("Soy un método inalcanzable desde
fuera")

>>> e = Ejemplo()
>>> e.__atributo_privado
Traceback (most recent call last):

File "<pyshell#106>", line 1, in <module>
e.__atributo_privado
AttributeError: 'Ejemplo' object has no attribute
'__atributo_privado'
```

Internamente la clase sí puede acceder a sus atributos y métodos encapsulados, el truco consiste en crear sus equivalentes "publicos":

```
>>> class Ejemplo:
    __atributo_privado = "Soy un atributo inalcanzable
desde fuera"
    def __metodo_privado(self):
        print("Soy un método inalcanzable desde
fuera")
    def atributo_publico(self):
        return self.__atributo_privado
    def metodo_publico(self):
        return self.__metodo_privado()

>>> e = Ejemplo()
>>> e.atributo_publico()
'Soy un atributo inalcanzable desde fuera'
>>> e.metodo_publico()
Soy un método inalcanzable desde fuera
>>>
```

Artillería de Ejercicios Propuestos POO

1. Queremos mantener una colección de los libros que hemos ido leyendo, poniéndoles una calificación según nos haya gustado más o menos al leerlo. Para ello, crear la clase Libro, cuyos atributos son el título, el autor, el número de páginas y la calificación que le damos entre 0 y 10. Crear los métodos típicos para poder insertar y obtener todos los libros insertados, eliminar y actualizar además mostrar por pantalla los libros con la mayor y menor calificación
2. Se quiere crear una clase Cuenta la cual se caracteriza por tener asociado un número de cuenta, DNI del cliente y nombre del cliente y un saldo disponible. Además, se puede consultar el saldo disponible en cualquier momento, recibir abonos y pagar recibos.
3. Crear una clase Rectángulo, con atributos base y altura. Crear también el constructor de la clase y los métodos necesarios para calcular el área y el perímetro. Crear varios objetos a partir de esa clase que pruebe varios rectángulos y muestre por pantalla sus áreas y perímetros.
4. Se conoce de un alumno: cédula, nombre y tres notas parciales (nota1, nota2, nota3). El programa debe imprimir: cédula, nombre, nota final e indique con un mensaje si el alumno aprobó (nota final ≥ 11) o no aprobó (nota final < 10) la asignatura.
5. Crear una clase llamada deportista que contenga los siguientes atributos: nombre, equipo, dorsal, puesto y cantidad de goles, crear los métodos para insertar deportistas y actualizar, eliminar, además un ranking de pichiche



Listas Recorridas

```
calificaciones = [100, 100, 90, 40, 80, 100, 85, 70, 90, 65, 90, 85, 50.5]
```

```
def print_calificaciones(calificaciones):
    #Imprime la lista de calificaciones
    for calificacion in calificaciones:
        print calificacion
```

```
def calificaciones_sum(lista):
    #Suma la calificaciones
    suma = 0.0
    for i in lista:
        suma += i

    return suma
```

```
def calificaciones_promedio(lista):
    #Calcula el promedio de las calificaciones
    suma = calificaciones_sum(lista)
    return suma / len(lista)
```

```
#print calificaciones No se pide, pero sino se imprime no nos
deja pasar de módulo.
print (calificaciones)
print (print_calificaciones(calificaciones))
print (calificaciones_sum(calificaciones))
print (calificaciones_promedio(calificaciones))
```



Diccionarios

```
mi_diccionario =
{'Venezuela':'Caracas','Alemania':'Berlin','Francia':'Paris'}
print (mi_diccionario['Venezuela'])
print (mi_diccionario)
```

```
# agregar un elemento al diccionario
mi_diccionario['Italia'] = 'Lisboa'
print (mi_diccionario)
```

```
# modificar un elemento del diccionario
mi_diccionario['Italia'] = 'Roma'
print (mi_diccionario)
```

```
# eliminar un elemento del diccionario
del mi_diccionario['Francia']
print (mi_diccionario)
```

```
tupla = ['Venezuela','España','Alemania']
diccionario =
{tupla[0]:'Caracas',tupla[1]:'Madrid',tupla[2]:'Berlin'}
print (diccionario)
```

```
# acceder a un elemento del diccionario
print (diccionario[tupla[0]])
print (diccionario['Venezuela'])
```

```
# diccionario almacena una tupla
# diccionario dentro de un diccionario con una tupla
jugador =
{'Nombre':'Michael','Apellido':'Jordan','Equipo':'Chicago','Anillos':
{'Temporadas':[1991,1992,1993,1996,1997,1998]}}
print (jugador)
print (jugador['Equipo'])
print (jugador['Anillos'])
```

```
# métodos para los diccionarios
# método keys me devuelve todas las claves del diccionario
print (jugador.keys())
```

```
# método values me devuelve los valores del diccionario
print (jugador.values())
```

```
# método len me devuelve la longitud del diccionario
print (len(jugador))
```




Challenger I

```
class Estudiante:
    def __init__(self,nom,eva,cal,com,fot,fereg):
        self.nombre = nom
        self.evaluacion = eva
        self.calificacion = cal
        self.comentario = com
        self.foto = fot
        self.fechareg =fereg
    def __str__(self):
        return '{} {} {} {} {} {}'
        .format(self.nombre,self.evaluacion,self.calificacion,self.com
entario,
        self.foto,self.fechareg)

class Estudiantes_Metodos:
    #Constructor
    def __init__(self):
        self.estudiantes = []

    def agregar(self,estudiantes):
        self.estudiantes.append(estudiantes)

    def __str__(self):
        cadena = "Total de
Estudiantes:"+str(len(self.estudiantes))
        for est in self.estudiantes:
            cadena += "\n" + str(est)
        return cadena

from estudiante import *

em = Estudiantes_Metodos()

def registrarestudiantes():
    print("Registro de Estudiantes")
    nombre = input("El Nombre:")
    evaluacion = input("La Evaluacion:")
    calificacion = input("La Calificacion:")
    comentario = input("El Comentario:")
    foto = input("La Foto:")
    fehareg = input("La Fecha Registro:")
    nuevoestudiante =
Estudiante(nombre,evaluacion,calificacion,comentario,foto,fe
chareg)
    em.agregar(nuevoestudiante)

def mostrarestudiantes():
    print(em)
    print("Carga Exitosa!!!")

def acumular():
```

```
print(em.acumular)
```

```
def menu():
    op = 0
    salir = 6
    while op != salir:
        print("Menu")
        print("*****")
        print("1-Registrar Estudiante")
        print("2-Listar Estudiante")
        print("6-Salir")
        op = input("Digite Opción:")
        if op == "1":
            registrarestudiantes()
        elif op == "2":
            mostrarestudiantes()
        elif op == "3":
            acumular()
        else :
            exit()
    menu()
```