# Curso de Python

Clase 03

# ¡Bienvenidxs!

# Contenido

- Programación orientada a objetos
  - Creación de clases, atributos y métodos
  - Decorador @property
  - Herencia
  - Polimorfismo
- Archivos
  - Entrada y salida
  - Archivos de tipo CSV y uso de fechas y horas.

# Pre-work

## Platzi: Curso de Python

3. **Estructuras de datos**
    - Manipulación de archivos en Python 3
4. **Uso de objetos y módulos**
    - ¿Qué es la programación orientada a objetos?
    - Programación orientada a objetos en Python

# CARPETAS DE TRABAJO SUGERIDA

Antes de iniciar a realizar programas, se recomienda crear una estructura de carpetas para esta clase de la siguiente manera:

```
Curso-de-Python
    └── Clase-03
        └── mi-programa.py
```

`Curso-de-Python`
: Carpeta que contendrá todo los archivos del curso

`Clase-03`
: Carpeta que contiene los archivos sólo para la Clase 03

`mi-programa.py`
: Archivo con terminación en .py que contendrá el programa en Python a ejecutar.

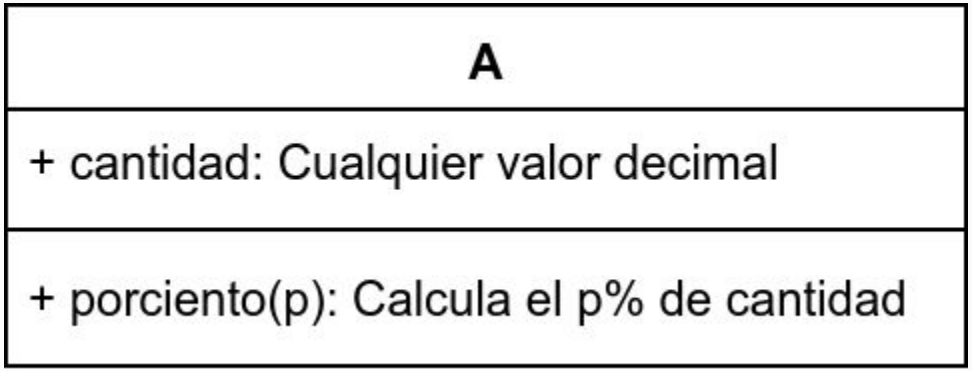
# Programación orientada a objetos

# Clases

Las clases son la manera de construir nuevos objetos en Python con sus datos o atributos y sus funciones o métodos.

Los objetos nos permiten representar objetos de nuestro entorno como puede ser una casa, auto, gato, persona, producto, etc.

# Representación del objeto A:



# CLASES

```
In [1]: # Definiendo una clase con atributos y funciones o métodos

In [2]:  class A:
   ...:      """ Define la clase A """
   ...:      def __init__(self, cantidad):
   ...:          """ Inicializa la clase """
   ...:          self.cantidad = cantidad  # inicializa el atributo cantidad
   ...:

In [3]: # Creando instancia de la clase
In [4]: a = A(100)
In [5]: # Obteniendo el valor de cantidad
In [6]: a.cantidad
Out[6]: 100
In [5]: # Cambiando el valor de cantidad
In [6]: a.cantidad = 200
In [6]: a.cantidad
Out[6]: 200
```

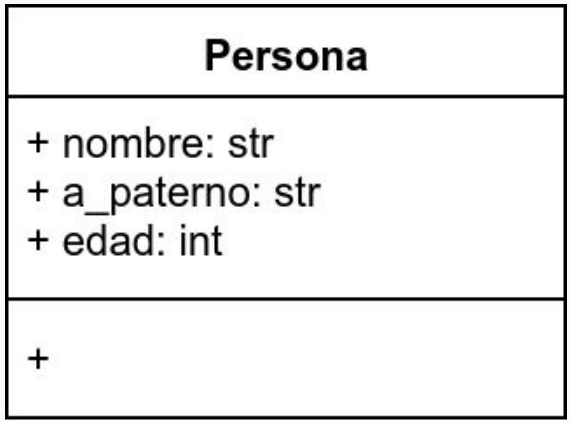
# CLASES

```
In [1]: # Definiendo una clase con atributos y funciones o métodos
In [2]:  class A:
   ...:     """ Define la clase A """
   ...:     def __init__(self, cantidad):
   ...:         """ Inicializa la clase """
   ...:         self.cantidad = cantidad  # inicializa el atributo cantidad
   ...:
   ...:     def porciento(self, p):
   ...:         “”” Calcula y regresa el porciento p% de valor “””
   ...:         return self.valor * p / 100

In [3]: # Creando instancia de la clase
In [4]: a = A(200)

In [7]: # Ejecutando la función o método
In [8]: a.porciento(30)  # Calculando el 30% de 200
Out[8]: 60.0
```

# LISTA DE CLASES DE PERSONAS

**Nombre del script:** `listas-de-personas.py`

**Descripción:**
El script deberá de crear e imprimir una lista de tres personas donde cada persona incluya nombre, apellido paterno y edad, además cada persona deberá de ser una instancia de la clase `Persona()`. Se debe hacer uso de las funciones `obtener_personas()` e `imprimir_personas()`.



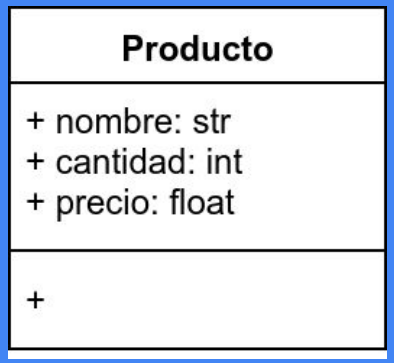
# LISTA DE CLASES DE PERSONAS

**Ejemplo de ejecución:**

```
Clase-03 $ python listas-de-personas.py
Nombre          Apellido         Edad
-------------------------------------
Hugo            Smith               8
Paco            Lorenz             28
Luis            Tesla              38
-------------------------------------

Clase-03 $
```

## ACTIVIDAD

Crea el script `listas-de-productos.py` para que imprima una lista de 3 productos, donde cada producto es una clase con los atributos nombre, cantidad y precio




Ejemplo de ejecución:

```
Clase-03 $ python lista-de-productos.py
Nombre          Cantidad Precio
-----------------------------------
Caja chica             5     100.00
Caja mediana           3     185.00
Caja grande            1     299.00
-----------------------------------

Clase-03 $
```

# Decorador @property

Un decorador es una función que extiende las capacidades de otra función sin modificar esta última.

El decorador @property permite que un método se comporte como un atributo de sólo lectura.

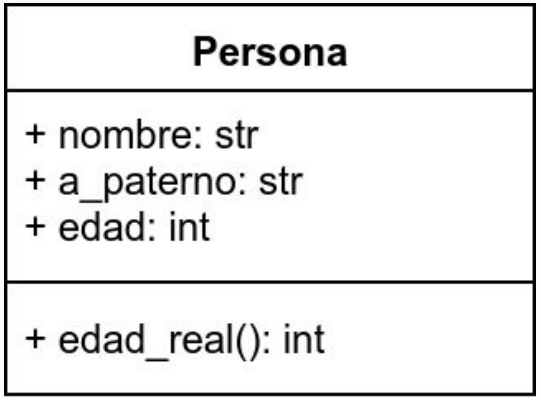
# DECORADOR @PROPERTY

```
In [1]: # Definiendo una clase A cuyo método suma() se comporta como un atributo
In [2]:  class A:
   ...:     """ Define la clase A """
   ...:     def __init__(self, x, y):
   ...:         """ Inicializa la clase y los atributos """
   ...:         self.x = x
   ...:         self.y = y
   ...:
   ...:     @property
   ...:     def suma(self):
   ...:         “”” Calcula y regresa la suma de x+y “””
   ...:         return self.x + self.y

In [3]: # Creando instancia de la clase
In [4]: a = A(100, 200)
In [5]: # Obteniendo la suma
In [6]: a.suma  # Se usa como atributo constante no como función
Out[6]: 300
```

# LISTA DE CLASES DE PERSONAS

**Nombre del script:** `listas-de-personas.py`

**Descripción:**
Suponer que en general una persona resta 5 años de su edad real, por lo que es necesario crear el método edad_real() que regresará el valor de la edad más 5, pero además le agregaremos el decorador @property para usar edad_real() como un atributo más y no como un método. Realizar las modificaciones en el script indicado.



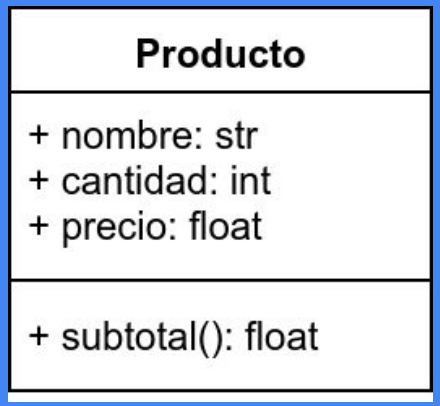
# LISTA DE CLASES DE PERSONAS

**Ejemplo de ejecución:**

```
Clase-03 $ python listas-de-personas.py
Nombre          Apellido         Edad Edad Real
-----------------------------------------------
Hugo            Smith               8        13
Paco            Lorenz             28        33
Luis            Tesla              38        43
-----------------------------------------------

Clase-03 $
```

## ACTIVIDAD

Modifica el script `listas-de-productos.py` para que imprima ademas de los 3 atributos (nombre, cantidad y precio), el atributo **subtotal** haciendo uso del decorador @property.




Ejemplo de ejecución:

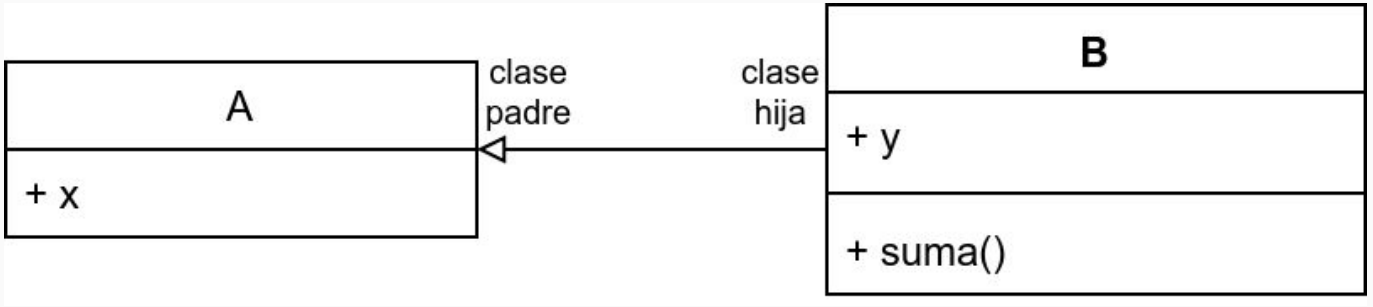
```
Clase-03 $ python lista-de-productos.py
Nombre          Cantidad Precio     Subtotal
-----------------------------------------------
Caja chica             5     100.00     500.00
Caja mediana           3     185.00     555.00
Caja grande            1     299.00     299.00
-----------------------------------------------
                              Total    1354.00

Clase-03 $
```

# Herencia

Herencia es el proceso por el que una clase toma los atributos y métodos de otra. La nueva clase formada es llamada clase hija y la clase de la que deriva se conoce como clase padre.

# HERENCIA

Representación de la herencia en un diagrama de clases:



# HERENCIA

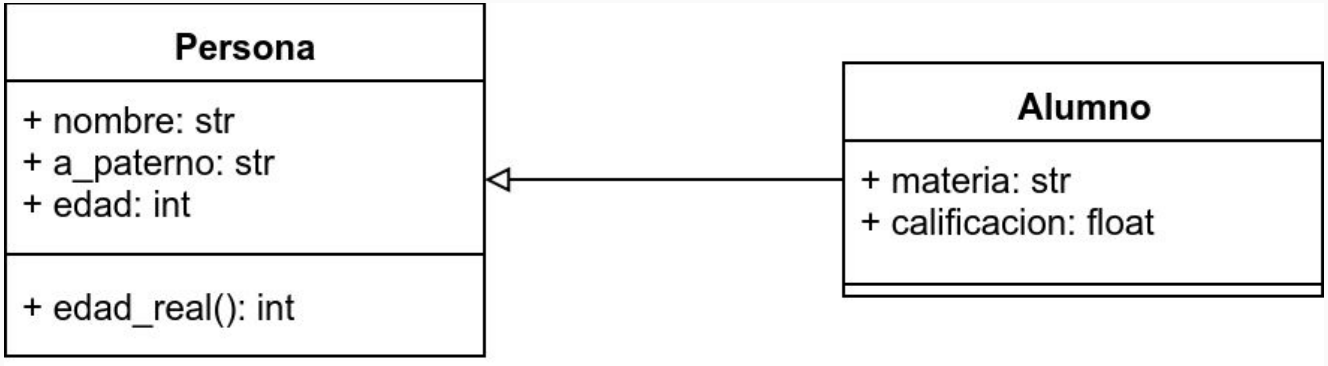
```
In [1]: # Definición de la clase A y la clase hija B
In [2]: class A:
   ...:     def __init__(self, x):
   ...:         self.x = x
In [3]: class B(A):
   ...:     def __init__(self, x, y):
   ...:         A.__init__(self, x)
   ...:         self.y = y
   ...:     def suma(self):
   ...:         return self.x + self.y

In [4]: b = B(5, 10)
In [5]: b.x
Out[5]: 5
In [6]: b.y
Out[6]: 10
In [6]: b.suma()
Out[6]: 15
```

# HERENCIA - CLASE HIJA ALUMNO

**Nombre del script:** `lista-alumnos.py`

**Descripción:**
Usar el script lista-personas.py como base para crear la clase hija Alumno(Persona) que agrega los atributos materia y calificación. Crear tres alumnos e imprimir la lista.

**Diagrama de clases:**



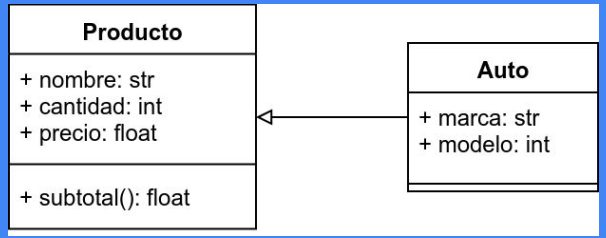
# HERENCIA - CLASE HIJA ALUMNO

**Ejemplo de ejecución:**

```
Clase-03 $ python lista-alumnos.py

Nombre          Apellido         Edad Edad Real Materia         Calificación
-------------------------------------------------------------------------
Hugo            Smith               8        13 Curso de Python          5.9
Paco            Lorenz             28        33 Curso de Python          8.0
Luis            Tesla              38        43 Curso de Python          9.9
-------------------------------------------------------------------------

Clase-03 $
```

## ACTIVIDAD

Crear el script `lista-autos.py` para que imprima la lista de 3 autos, haciendo uso de la clase Auto(Producto) que además cuenta con los atributos **marca** y **modelo**



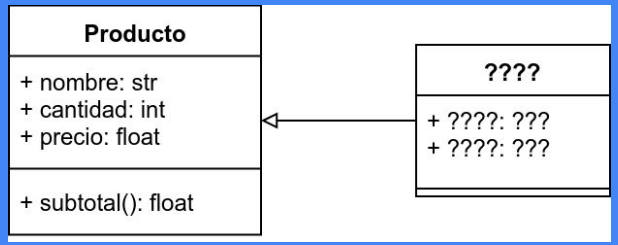

Ejemplo de ejecución:

```
Clase-03 $ python lista-autos.py

Marca      Nombre          Modelo Precio
-------------------------------------------
VW         Vocho             2000   10000.00
Seat       Cordoba           2010  185000.00
Chevrolet  Camaro            2018  299000.00
-------------------------------------------
                           Total  494000.00

Clase-03 $
```

CLASE HIJA ????()

## ACTIVIDAD

Crear el script `lista-????.py` para que imprima la lista de 3 productos que tengan una clase hija de Productos cuya definición sería class ??????(Productos) y agrega cuando menos un atributo que sea exclusivo de este tipo de producto como en el caso de la clase Auto().




Ejemplo de ejecución:

```
Clase-02 $ python lista-????.py

Nombre          Color  Cantidad Precio
-----------------------------------------
...             Blanco        2    1000.00
...             Perla         3    1500.00
...             Café          1    2500.00
-----------------------------------------
                           Total    9000.00
```

# Polimorfismo

Es una propiedad de las clases que es utilizada cuando se tienen métodos con el mismo nombre entre clases o subclases y permite a un método producir distintos resultados según la clase del objeto.

# POLIMORFISMO

```
# Definición de la clase A y la clase B
In [1]: class A:
    ...:     def saludo(self):
    ...:         print("Soy la primera clase, soy la A")
In [2]: class B:
    ...:     def saludo(self):
    ...:         print("Soy una Buena clase, soy la B")

# Instanciando las clases
In [3]: a = A()
In [4]: b = B()

# Creando la lista [a, b] cuyos elementos son dos clases distintas
In [5]: for objeto in [a, b]:
    ...:     # pero el método saludo() existe en ambas clases
    ...:     objeto.saludo()
Soy la primera clase, soy la A
Soy una Buena clase, soy la B
```

# CLASE HIJA ALUMNOS

**Nombre del script:** `lista-personas-alumnos.py`

**Descripción:**
El script deberá instanciar 3 personas y 3 alumnos y luego imprimirlos en una tabla en la salida estándar haciendo uso de sólo una función imprime_lista()

**Ejemplo de ejecución con las clases sin el método str():**

```
Clase-02 $ python lista-personas-alumnos.py
--------------------------------------------------------------------------------
<__main__.Persona object at 0x7fbf0f1c5d30>
<__main__.Persona object at 0x7fbf0f1c5d68>
<__main__.Persona object at 0x7fbf0f1c5dd8>
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
<__main__.Alumno object at 0x7fbf0f1c5f60>
<__main__.Alumno object at 0x7fbf0f1c5f98>
<__main__.Alumno object at 0x7fbf0f1c5fd0>
--------------------------------------------------------------------------------
```

# CLASE HIJA ALUMNOS

**Ejemplo de ejecución con el método str() sólo en la clase Persona:**

```
Clase-02 $ python lista-personas-alumnos.py
--------------------------------------------------------------------------------
Hugo            Smith             8       13
Paco            Lorenz           28       33
Luis            Tesla            38       43
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
Hugo            Smith             8       13
Paco            Lorenz           28       33
Luis            Tesla            38       43
--------------------------------------------------------------------------------
```

**Ejemplo de ejecución con el método str() en ambas clases:**

```
Clase-02 $ python lista-personas-alumnos.py
--------------------------------------------------------------------------------
Hugo            Smith             8       13
Paco            Lorenz           28       33
Luis            Tesla            38       43
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
Hugo            Smith             8       13 Curso de Python          5.9
Paco            Lorenz           28       33 Curso de Python          8.0
Luis            Tesla            38       43 Curso de Python          9.9
--------------------------------------------------------------------------------
```

## ACTIVIDAD

Crear el script `lista-productos-autos.py` para que imprima la lista de 3 **productos** con atributos **nombre, cantidad y precio** y 3 **autos** con atributos **nombre, cantidad, precio, marca y modelo** haciendo uso de sólo una función llamada `imprime_productos()`

**TIP:** Recordar definir y hacer uso del método `str()`

**RETO:** Imprimir el total de cada conjunto de productos

Ejemplo de ejecución:

```
Clase-02 $ python lista-productos-autos.py
-------------------------------------------------------------------
Caja chica              5     100.00     500.00
Caja mediana            3     185.00     555.00
Caja grande             1     299.00     299.00
-------------------------------------------------------------------
                              Total:    1354.00


-------------------------------------------------------------------
VW Vocho (2000)                 1   10000.00   10000.00
Seat Cordoba (2010)             1  185000.00  185000.00
Chevrolet Camaro (2018)         1  299000.00  299000.00
-------------------------------------------------------------------
                                      Total:  494000.00
```

# Archivos en Python

# Entrada y Salida

Los archivos, es el medio por el cual la información se hace persistente por medio de un sistema de archivos y para poder usarlos es necesario contar con las operaciones básicas de **abrir**, **leer**, **escribir** y **cerrar** flujos de datos.

# LECTURA DE ARCHIVOS

```
# Apertura e impresión de las líneas de un archivo de texto
In [1]: fa = open("lista-alumnos.py")  # apertura de texto y para lectura
In [2]: for linea in fa:
   ...:     print(linea)
#!/usr/bin/env python

# -*- coding: utf-8
…

# Cerrando el archivo
In [3]: fa.close()

# Versión con with que cierra el archivo
In [4]: with open("lista-alumnos.py", "r") as fa:
   ...:     texto = fa.read()
   ...: print(texto)
```

# ESCRITURA DE ARCHIVOS

```
# Creando un archivo de texto con 3 alumnos
In [1]: with open("alumnos.txt", "w") as fa:
   ...:     fa.write("Alumno1, Apellido1, Materia1, Calif1\n")
   ...:     fa.write("Alumno2, Apellido3, Materia2, Calif2\n")
   ...:     fa.write("Alumno3, Apellido3, Materia3, Calif3\n")

# Mostrar el contenido del archivo alumnos.txt en Linux / Mac
In [2]: !cat alumnos.txt
Alumno1, Apellido1, Materia1, Calif1
Alumno2, Apellido2, Materia2, Calif2
Alumno3, Apellido3, Materia3, Calif3

# Mostrar el contenido del archivo alumnos.txt en Windows
In [3]: !type alumnos.txt
Alumno1, Apellido1, Materia1, Calif1
Alumno2, Apellido2, Materia2, Calif2
Alumno3, Apellido3, Materia3, Calif3
```

# ARCHIVOS PERSISTENTES

**Nombre del script:** `personas-alumnos-archivos-txt.py`

**Descripción:**
Agregar al script `lista-personas-alumnos.py` la función **`guardar_lista()`** que reciba una lista de alumnos o personas y que guarde la lista en el archivo de texto llamado `lista.txt`

**Ejemplo de ejecución:**

```
Clase-03 $ python personas-alumnos-archivos-txt.py

Clase-03 $ cat lista.txt  # type lista.txt para usuarios de windows
--------------------------------------------------------------------------------
Hugo            Smith               8        13
Paco            Lorenz             28        33
Luis            Tesla              38        43
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
Hugo            Smith               8        13 Curso de Python          5.9
Paco            Lorenz             28        33 Curso de Python          8.0
Luis            Tesla              38        43 Curso de Python          9.9
--------------------------------------------------------------------------------

Clase-03 $
```

## ACTIVIDAD

Crear el script `productos-autos-archivos-txt.py` para que guarde la lista de 3 **productos** con atributos **nombre, cantidad y precio** y 3 **autos** con atributos **nombre, cantidad, precio, marca y modelo** en un archivo de texto llamado `productos.txt` haciendo uso de sólo una función llamada `guardar_productos()`

Ejemplo de ejecución:

```
Clase-03 $ python productos-autos-archivos-txt.py

Clase-03 $ cat productos.txt
------------------------------------------------------------------
Caja chica              5     100.00     500.00
Caja mediana            3     185.00     555.00
Caja grande             1     299.00     299.00
------------------------------------------------------------------
                              Total:    1354.00
------------------------------------------------------------------
VW Vocho (2000)                   1   10000.00   10000.00
Seat Cordoba (2010)               1  185000.00  185000.00
Chevrolet Camaro (2018)           1  299000.00  299000.00
------------------------------------------------------------------
                                        Total:  494000.00
```

# Archivos de tipo CSV

# El formato CSV

Un archivo en formato csv (*comma-separated values)* es un archivo de texto que almacena los datos en forma de columnas, separadas por coma y las filas se distinguen por saltos de línea y son usados principalmente para almacenar o intercambiar la información entre un sistema y otro.

# EL MÓDULO CSV

```
# Importando el módulo CSV
In [1]: import csv
# Abrir el archivo para escritura
# Crear una variable asociada con el módulo CSV que nos permita escribir
# Escribir cada registro
In [2]: with open("ejemplo.csv", "w") as fcsv:
   ...:     csv_writer = csv.writer(fcsv)
   ...:     csv_writer.writerow(["Nombre1", "Apellido1", "Email1"])
   ...:     csv_writer.writerow(["Nombre2", "Apellido2", "Email2"])
   ...:     csv_writer.writerow(["Nombre3", "Apellido3", "Email3"])

# Leyendo el contenido del archivo ejemplo.csv
In [3]: with open("ejemplo.csv") as fcsv:
   ...:     csv_reader = csv.reader(fcsv)
   ...:     for registro in csv_reader:
   ...:         print(registro)
["Nombre1", "Apellido1", "Email1"]
["Nombre2", "Apellido2", "Email2"]
["Nombre3", "Apellido3", "Email3"]
```

# FECHAS y HORAS en PYTHON

```
# Importando el módulo datetime
In [1]: import datetime
# Obteniendo la fecha actual
In [2]: hoy = datetime.datetime.today()
In [3]: hoy
Out[3]: datetime.datetime(2019, 2, 11, 16, 19, 3, 479817)

# Formateando fechas a str
In [4]: hoy.strftime("%Y-%m-%d %H:%M")
Out[4]: '2019-02-11 16:19'
In [5]: hoy.strftime("%b %d %Y %H:%M")
Out[6]: 'Feb 11 2019 16:19'

# Atributos de una fecha-tiempo
In [5]: hoy.year
Out[6]: 2019
In [5]: hoy.day
Out[6]: 11
In [5]: hoy.hour
Out[6]: 16
```

# LISTA DE ARCHIVOS DEL DIRECTORIO ACTUAL - MÓDULOS CSV Y OS

**Nombre del script:** `archivos2csv.py`

**Descripción:**
El script deberá de crear el archivo `archivos.csv` con la lista de archivos del directorio actual ordenados en base al tamaño en bytes, los más grandes primero, deberá incluir sólo los archivos con extensión `.py`, deberá hacer uso de funciones, hacer uso de una clase Archivo con los atributos nombre, tamaño en bytes y fecha de modificación, finalmente debe poder usarse como módulo. Se sugiere hacer uso del módulo `archivos` creado con anterioridad en el paquete `utilerias`.

El diagrama de la clases sería el siguiente:

# LISTA DE ARCHIVOS DEL DIRECTORIO ACTUAL - MÓDULOS CSV Y OS

**Ejemplo de ejecución:**

```
Clase-03 $ python archivos2csv.py

Clase-03 $ cat archivos.csv
personas-alumnos-archivos-txt.py,3969,Feb 11 2019 20:47
lista-de-personas-alumnos.py,3326,Feb 11 2019 20:47
lista-de-alumnos.py,2860,Feb 11 2019 20:47
lista-de-personas.py,2270,Feb 11 2019 20:47
lista-de-personas.py,1948,Feb 11 2019 20:47
archivos2csv.py,1070,Feb 11 2019 22:00

Clase-03 $
```

PRODUCTOS PERSISTENTES EN CSV

**ACTIVIDAD**

Crear el script `productos2csv.py` para que guarde 3 **productos** con atributos **nombre, cantidad, precio** y **subtotal** en el archivo `productos.csv`. Hacer uso de la función `guarda_productos_csv()` y la clase **Producto**

**TIP:** Basarse el la última versión del script `lista-de-productos.py`

**RETO:** Agregar el total como una fila más al archivo `productos.csv`

Intente abrir el archivo resultante en una hoja de cálculo.

Ejemplo de ejecución:

```
Clase-03 $ python productos2csv.py
El archivo productos.csv ha sido creado!

Clase-03 $ cat productos.csv
Caja chica,5,100.0,500.0
Caja mediana,3,185.0,555.0
Caja grande,1,299.0,299.0
,,,1354.0

Clase-03 $
```

| | A | B | C | D |
|---|---|---|---|---|
| 1 | Caja chica | 5 | 100 | 500 |
| 2 | Caja mediana | 3 | 185 | 555 |
| 3 | Caja grande | 1 | 299 | 299 |
| 4 | | | | 1354 |

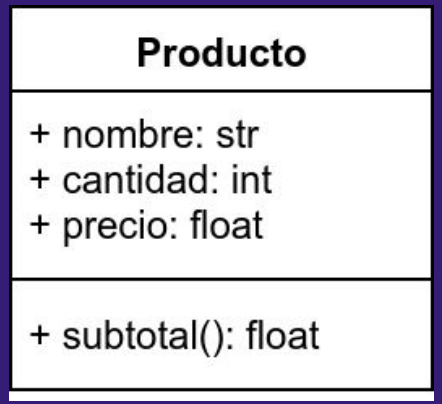
**Lista de scripts a crear:**

```
agrega-producto.py
nota-de-venta.py
productos.csv
nota.csv
notas/
|- __init__.py
|- entrada.py
|- producto.py
|- salida.py
```

**Descripción:**
Modificar los script del proyecto final de la Clase-02 para que la nota de venta se guarde en un archivo llamado `nota.csv` haciendo uso del objeto Producto y definiendo las funciones de entrada y salida en sus respectivos módulos.

Adicionalmente se tiene que manejar persistencia de los productos en el archivo `productos.csv` haciendo uso de una clase llamada `Producto()` y el script `agrega-productos.py`.

El diagrama de la clase Producto es:

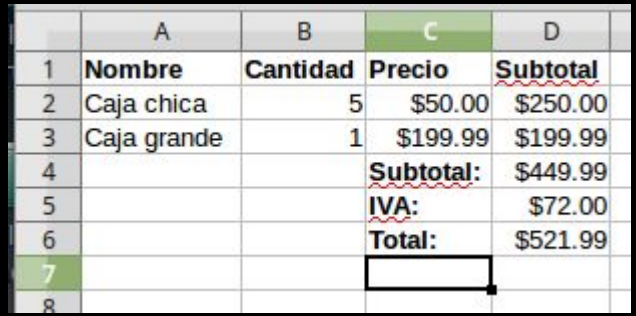


**Ejemplo de ejecución del script `agrega-producto.py`:**

```
Clase-03 $ python agrega-producto.py
Nombre del producto: Caja chica
Cantidad [1]: 5
Precio: 50.0
El producto Caja chica ha sido guardado en productos.csv!

Clase-03 $ python agrega-producto.py
Nombre del producto: Caja grande
Cantidad [1]:
Precio: 199.99
El producto Caja grande ha sido guardado en productos.csv!

Clase-03 $ cat productos.csv
Caja chica,5,50.0,250.0
Caja grande,1,199.99,199.99

Clase-03 $
```

**Ejemplo de ejecución del script `nota-de-venta.py`:**

```
Clase-03 $ python nota-de-venta.py
Se han leído 2 productos de productos.csv!
Se ha creado la nota de venta en el archivo nota.csv!

Clase-03 $ cat nota.csv
Nombre,Cantidad,Precio,Subtotal
Caja chica,5,50.0,250.0
Caja grande,1,199.99,199.99
,,Subtotal:,449.99
,,IVA:,72.0
,,Total:,521.99

Clase-03 $
```



| | A | B | C | D |
|---|---|---|---|---|
| 1 | **Nombre** | **Cantidad** | **Precio** | **Subtotal** |
| 2 | Caja chica | 5 | $50.00 | $250.00 |
| 3 | Caja grande | 1 | $199.99 | $199.99 |
| 4 | | | **Subtotal:** | $449.99 |
| 5 | | | **IVA:** | $72.00 |
| 6 | | | **Total:** | $521.99 |
| 7 | | | | |
| 8 | | | | |