

Introducción a la programación con Python

Programación Orientada a Objetos y Clases

Alexis Rodríguez

Marcel Morán C

Esquema

- Un programa de recetas
- ¿Qué es un OOP?
- ¿Qué es una Clase?
- Sintaxis de Clases

Un programa de recetas



Un programa de recetas



```
recetas = {}
```

```
recetas["Encebollado"] = ["atun", "tomates", "apio"]
```

```
recetas["Bolon"] = ["verdes", "queso", "pollo"]
```

```
recetas["Guatita"] = ["arroz", "aguacate", "panza de res"]
```

```
recetas["Hornado"] = ["papas", "carne de chanco", "mote"]
```

```
print(recetas["Encebollado"])
```

```
>>> ['atun', 'tomates', 'apio']
```

Un programa de recetas



```
recetas = {}
```

```
ingredientes = {"atún":2, "tomates":4 , "apio":2}
```

```
recetas["Encebollado"] = ingredientes
```



```
ingredientes = {"papas":2, "carne de chanco":2 , "mote":10}
```

```
recetas["Hornado"] = ingredientes
```

```
print(recetas["Encebollado"])
```



```
>>> {'atún': 2, 'tomates': 4, 'apio': 2}
```

Un programa de recetas



```
recetas = {}  
ingredientes = {"atún":2, "tomates":4 , "apio":2}  
valor_nutricional_kcal = 443  
recetas["Encebollado"] = [ingredientes,valor_nutricional_kcal ]  
ingredientes = {"papas":2, "carne de chanco":2 , "mote":10}  
valor_nutricional_kcal = 480  
recetas["Hornado"] = [ingredientes,valor_nutricional_kcal ]  
print(recetas["Encebollado"])
```

```
[{'atún': 2, 'tomates': 4, 'apio': 2}, 443]
```

Un programa de recetas



```
def num_total_de_ingredientes(diccionario):  
    print("El numero total de ingredientes es",  
          sum(diccionario.values()))
```

```
recetas = {}  
ingredientes = { "atún":2, "tomates":4 , "apio":2}  
valor_nutricional_kcal = 443  
recetas["Encebollado"] = [ingredientes,valor_nutricional_kcal ]  
ingredientes = { "papas":2, "carne de chanco":2 , "mote":10}
```

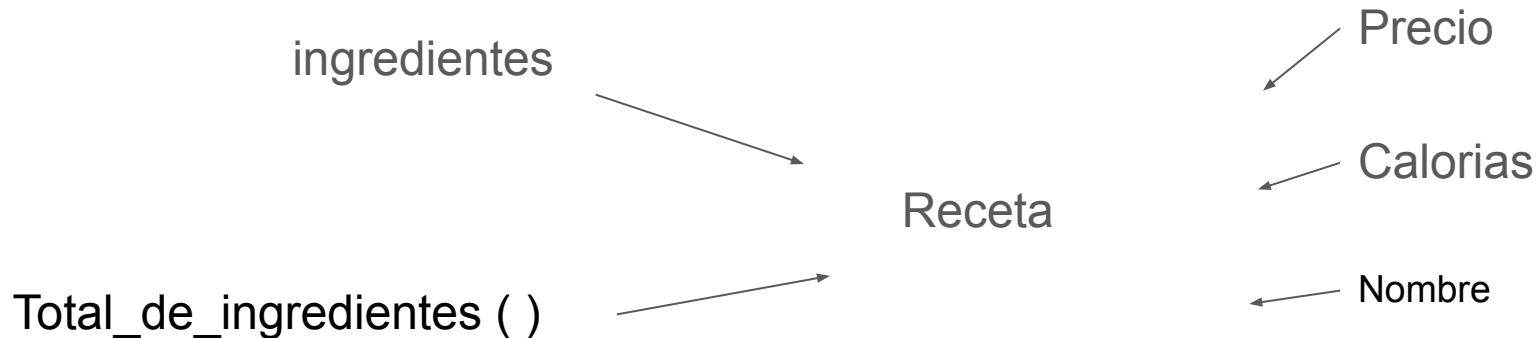
```
num_total_de_ingredientes(recetas[ "Encebollado"][0])
```

Programación Orientada a Objetos

- Es un paradigma de programación enfoca en descomponer tareas en Objetos
- La programación imperativa se centra en describir cómo se deben realizar las operaciones paso a paso
- Objetos contienen datos (atributos) e instrucciones(métodos)
- Encapsulación, Herencia y Poliformismo

Programación Orientada a Objetos

- Es un paradigma de programación enfoca en descomponer tareas en Objetos
- La programación imperativa se centra en describir cómo se deben realizar las operaciones paso a paso
- **Objetos contienen datos (atributos) e instrucciones(métodos)**
- **Objetos son creados con Clases**
- Abstracción, Encapsulación, Herencia y Poliformismo



Programación Orientada a Objetos

- Es un paradigma de programación enfoca en descomponer tareas en Objetos
- La programación imperativa se centra en describir cómo se deben realizar las operaciones paso a paso
- **Objetos contienen datos (atributos) e instrucciones(métodos)**
- **Objetos son creados con Clases**
- Abstracción, Encapsulación, Herencia y Poliformismo

`{"verdes":1, "queso":1 , "pollo":1}`

Total_de_ingredientes ()

3



3 \$

450

Bolon

Programación Orientada a Objetos

- Es un paradigma de programación enfoca en descomponer tareas en Objetos
- La programación imperativa se centra en describir cómo se deben realizar las operaciones paso a paso
- **Objetos contienen datos (atributos) e instrucciones(metodos)**
- **Objetos son creados con Clases**
- Abstracción, **Encapsulación**, Herencia y Poliformismo

**{"papas":2, "carne de
chanchito":2 , "mote":10}**



Total_de_ingredientes ()

14

3 \$



432



Hornado



Programación Orientada a Objetos

- Es un paradigma de programación enfoca en descomponer tareas en Objetos
- La programación imperativa se centra en describir cómo se deben realizar las operaciones paso a paso
- **Objetos contienen datos (atributos) e instrucciones(métodos)**
- **Objetos son creados con Clases**
- **Abstracción, Encapsulación, Herencia y Poliformismo**

```
{"papas":2, "carne de chanco":2  
, "mote":10}
```

Total_de_ingredientes ()

sum(ingredientes.values()) = 14



3 \$

432

Hornado

Programación Orientada a Objetos

- Es un paradigma de programación enfoca en descomponer tareas en Objetos
- La programación imperativa se centra en describir cómo se deben realizar las operaciones paso a paso
- Objetos contienen datos (atributos) e instrucciones(métodos)
- Objetos son creados con Clases
- **Abstracción, Encapsulación, Herencia y Poliformismo**



```
from datetime import date
```

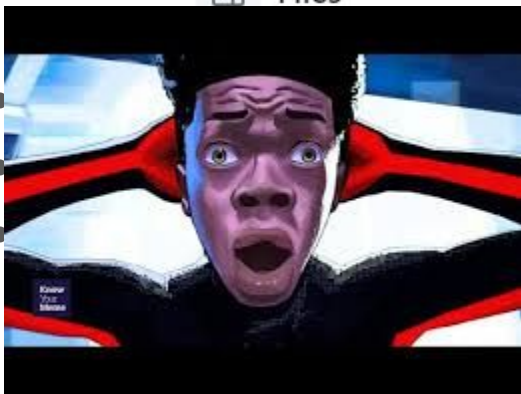
```
print("El día de hoy es ", date.today())
```

```
>>> El día de hoy es 14/03/2022
```

Program




- Es un pa
- La progr



Code

Issues


5k+

 Pull requests

1.7k

⏮ Actions

Projects 28

 Security

Insights

Files

cpython / Modules / _datetimemodule.c 

Yhg1s

Fix undefined behaviour in `datetime.time.fromisoformat()` (#11)

Code

Blame

7170 lines (6253 loc) · 228 KB

```
1  /* C implementation for the date/time type documented at
2     * https://www.zope.dev/Members/fdrake/DateTimeWiki/Front
3     */
4
```

```
print("El dia de hoy es ", date.today())
```

>>> El dia de hoy es 14/03/2022

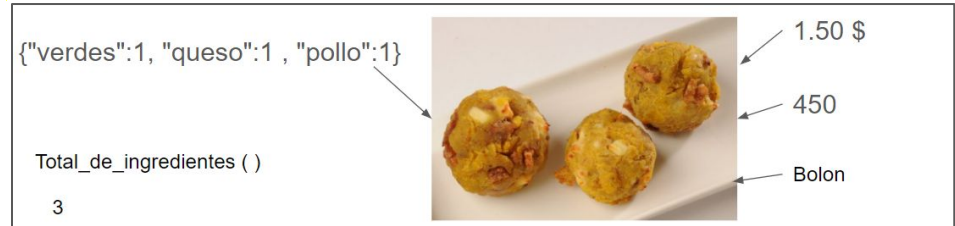
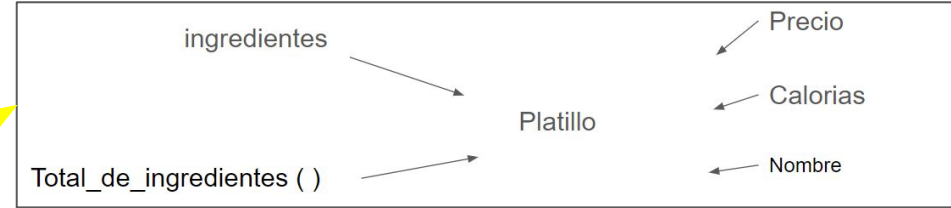
¿Qué es una Clase?

- Es un molde para la creación de objetos
- Contiene los atributos y métodos



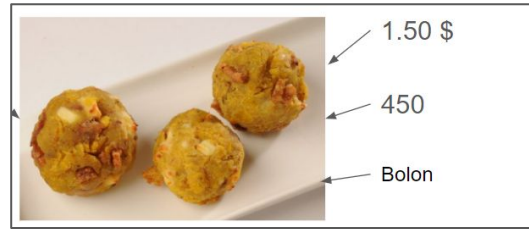
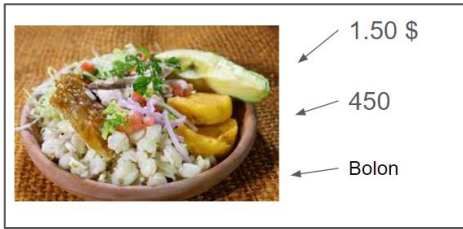
Clase

Objeto



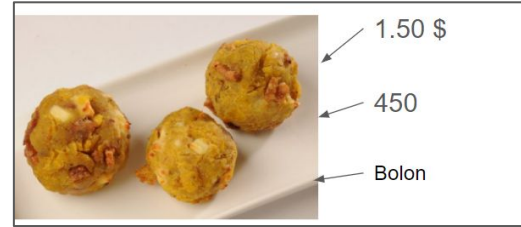
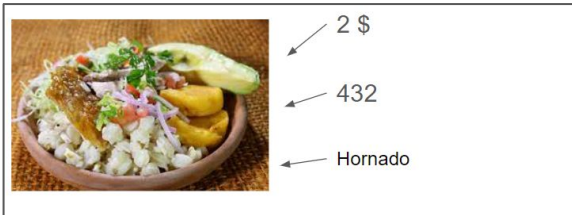
¿Qué es una Clase?

- Es un molde para la creación de objetos
- Contiene los atributos y métodos
- Objeto == instancia de una clase
- **Atributos de clase e instancia**
- Métodos de clase e instancia



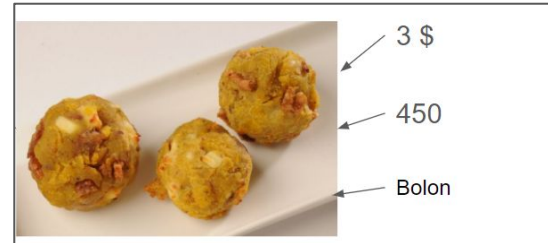
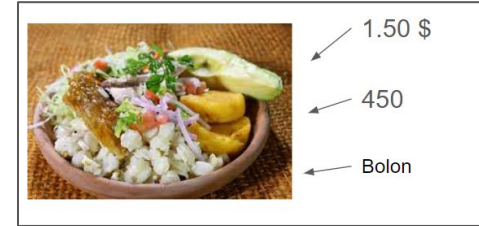
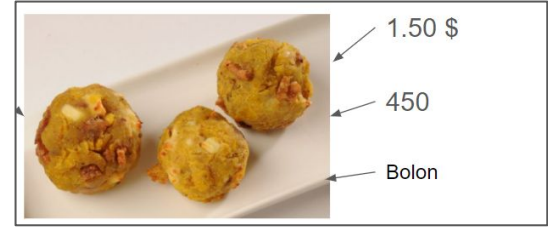
¿Qué es una Clase?

- Es un molde para la creación de objetos
- Contiene los atributos y métodos
- Objeto == instancia de una clase
- **Atributos de clase e instancia**
- Métodos de clase e instancia



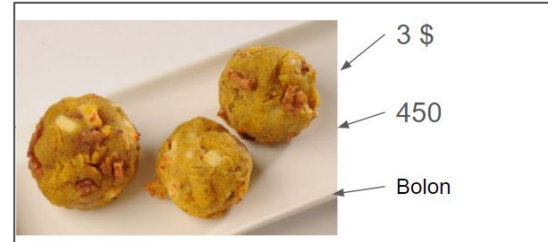
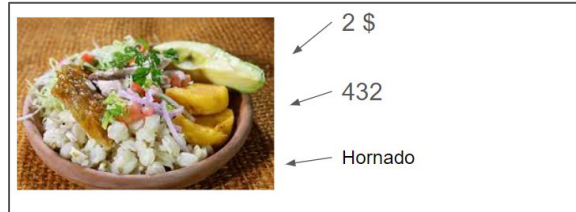
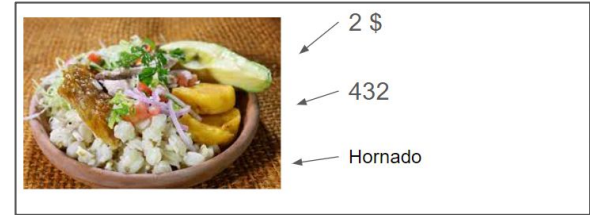
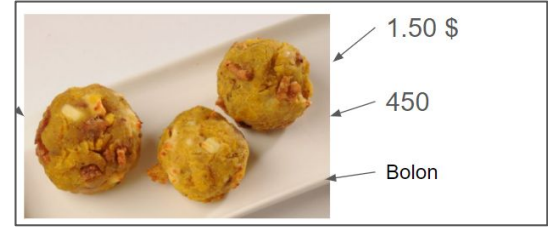
¿Qué es una Clase?

- Es un molde para la creación de objetos
- Contiene los atributos y métodos
- Objeto == instancia de una clase
- Atributos de clase e instancia
- **Métodos de clase e instancia**



¿Qué es una Clase?

- Es un molde para la creación de objetos
- Contiene los atributos y métodos
- Objeto == instancia de una clase
- Atributos de clase e instancia
- **Métodos de clase e instancia**



¿Qué es una Clase?

- Es un molde para la creación de objetos
- Contiene los atributos y métodos
- Objeto == instancia de una clase
- Atributos de clase e **instancia**
- Métodos de clase e **instancia**

Sintaxis de una Clase

- Se define una clase con las palabras **class Nombre**
- Constructor **__init__**(self, parameter1, parameter2)
- Instanciar con **Nombre**
- Metodos require self como parametro

class Receta:

```
def __init__(self, nombre, ingredientes, kcal):
```

```
    self.nombre = nombre
```

```
    self.ingredientes = ingredientes
```

```
    self.kcal = kcal
```

```
def num_total_de_ingredientes(self):
```

```
    print(self.nombre, ":Total de ingredientes: ", sum(self.ingredientes.values()))
```

Sintaxis de una Clase

```
class Receta:
```

```
def __init__(self, nombre, ingredientes, kcal):
```

```
    self.nombre = nombre
```

```
    self.ingredientes = ingredientes
```

```
    self.kcal = kcal
```

```
def num_total_de_ingredientes(self):
```

```
    print(self.nombre, ":Total de ingredientes: ", sum(self.ingredientes.values()))
```



Sintaxis de una Clase

```
ingredientes = {"atún":2, "tomates":4 , "apio":2}  
valor_nutricional_kcal = 443  
nombre_de_platillo = "Encebollado"  
encebollado = Receta(nombre_de_platillo,ingredientes,valor_nutricional_kcal)
```

```
nombre_de_platillo = "Hornado"  
ingredientes = {"papas":2, "carne de choncho":2 , "mote":10}  
valor_nutricional_kcal = 480  
Hornado = Receta(nombre_de_platillo,ingredientes,valor_nutricional_kcal)
```

Sintaxis de una Clase

- Se define una clase con las palabras **class Nombre**
- Constructor **__init__**(self, parameter1, parameter2)
- Instanciar con **Nombre**
- Metodos require self como parametro

```
ingredientes = {"atún":2, "tomates":4 , "apio":2}
```

```
valor_nutricional_kcal = 443
```

```
nombre_de_platillo = "Encebollado"
```

```
encebollado = Receta(nombre_de_platillo,ingredientes,valor_nutricional_kcal)
```

```
nombre_de_platillo = "Hornado"
```

```
ingredientes = {"papas":2, "carne de chancho":2 , "mote":10}
```

```
valor_nutricional_kcal = 480
```

```
Hornado = Receta(nombre_de_platillo,ingredientes,valor_nutricional_kcal)
```


Sintaxis de una Clase

- Se define una clase con las palabras **class Nombre**
- Constructor **__init__**(self, parameter1, parameter2)
- Instanciar con **Nombre**
- Metodos require self como parametro

```
>>> Encebollado :Total de ingredientes: 8
```

```
>>> Hornado :Total de ingredientes: 14
```

Sintaxis de una Clase

- Se define una clase con las palabras **class Nombre**
- Constructor **__init__**(self, parameter1, parameter2)
- Instanciar con **Nombre**
- Metodos require self como parametro

```
print(encebollado)
```

```
print(hornado)
```

```
>>> <__main__.Receta object at 0x00000>
```

```
>>><__main__.Receta object at 0x00000>
```

Sintaxis de una Clase

- Se define una clase con las palabras **class Nombre**
- Constructor **__init__(self, parameter1, parameter2)**
- Instanciar con **Nombre**
- Metodos require self como parametro
- **__str__** definir un mensaje para imprimir un objeto

class Receta:

.

```
def __str__(self):  
    self.num_total_de_ingredientes()  
    return "Ingredientes:" + str(self.ingredientes) + "\nCon kcal:" + str(self.kcal) + "\n"
```

Sintaxis de una Clase

- Se define una clase con las palabras **class Nombre**
- Constructor **__init__**(self, parameter1, parameter2)
- Instanciar con **Nombre**
- Metodos require self como parametro
- **__str__** definir un mensaje para imprimir un objeto

`print(encebollado)`

`print(hornado)`

```
(intro_python) C:\Users\Diego\Desktop\curso_de_python>python recetas.py
Encebollado
Total de ingredientes: 8
Ingredientes: {'atún': 2, 'tomates': 4, 'apio': 2}
Con kcal:443

Hornado
Total de ingredientes: 14
Ingredientes: {'papas': 2, 'carne de chancho': 2, 'mote': 10}
Con kcal:480
```

Sintaxis de una Clase

- Se define una clase con las palabras **class Nombre**
- Constructor **__init__**(self, parameter1, parameter2)
- Instanciar con **Nombre**
- Metodos require self como parametro
- **__str__** definir un mensaje para imprimir un objeto
- **from archivo import Class** para importar una clase de otro archivo

Conclusión

- POO o Programación orientada a objetos es paradigma de programación
- Un objeto se refiere a una combinación entre data e instrucciones
- Principios de objetos, encapsulacion
- Las clases son el molde de un objeto
- Sintaxis de clases `__init__`, `class`, `__str__`, `self`
- La creation de la clase Receta

Retroalimentación

- Para retroalimentación dirigirse al siguiente enlace <https://forms.gle/TYk5DonisBxt4GF27> .
- Déjanos saber qué podemos hacer para mejorar el curso

