# Intro a Patróns de deseño con Python: Exemplo práctico

## Introducción a Patróns & Python

M. Torre Castro[1]

[1]Python Coruña

Charlas Python Coruña, 2023

# Índice



# Outline



# Programación tradicional antes da POO

## O pasado imperativo...

- Antes da entrada en vigor das linguaxes Simula e Smalltalk, as linguaxes de programación eran principalmente imperativas
- Os programas eran secuencias de definicións de datos e instruccións que operaban sobre eles
- As linguaxes non proveen dunha ferramenta que permita separar o código en unidades conceptuais (hai certas excepcións)

# Programación tradicional antes da POO

## O pasado imperativo...

- Antes da entrada en vigor das linguaxes Simula e Smalltalk, as linguaxes de programación eran principalmente imperativas
- Os programas eran secuencias de definicións de datos e instruccións que operaban sobre eles
- As linguaxes non proveen dunha ferramenta que permita separar o código en unidades conceptuais (hai certas excepcións)

# Programación tradicional antes da POO

## O pasado imperativo...

- Antes da entrada en vigor das linguaxes Simula e Smalltalk, as linguaxes de programación eran principalmente imperativas
- Os programas eran secuencias de definicións de datos e instruccións que operaban sobre eles
- As linguaxes non proveen dunha ferramenta que permita separar o código en unidades conceptuais (hai certas excepcións)

# Que é a Programación Orientada a Obxetos?

## Cambiando o chip...

- Simula e Smalltalk nos ofrecen obxetos
- Un obxeto agrupa datos e operacións
- O obxeto ven definido por unha clase, que define a estructura do obxeto e permite facer instancias

# Que é a Programación Orientada a Obxetos?

## Cambiando o chip...

- Simula e Smalltalk nos ofrecen obxetos
- Un obxeto agrupa datos e operacións
- O obxeto ven definido por unha clase, que define a estructura do obxeto e permite facer instancias

# Que é a Programación Orientada a Obxetos?

## Cambiando o chip...

- Simula e Smalltalk nos ofrecen obxetos
- Un obxeto agrupa datos e operacións
- O obxeto ven definido por unha clase, que define a estructura do obxeto e permite facer instancias

# Algunhas consecuencias de usar POO

## As características básicas da POO

- Encapsulación: datos e operacións relacionados van xuntos e son accedidos soamente pola clase
- Abstracción: os conceptos complexos quedan ocultos no obxeto ou clase que os abarca e que nos provee de atributos e métodos faciles de manexar
- Herencia: As clases poden compartir información (atributos) e comportamento (métodos) e facer xerarquias
- Polimorfismo: Os obxetos poden ter diferente comportamento según o contexto e tamén poden ser tratados igual ainda que os tipos sexan distintos

# Algunhas consecuencias de usar POO

## As características básicas da POO

- Encapsulación: datos e operacións relacionados van xuntos e son accedidos soamente pola clase
- Abstracción: os conceptos complexos quedan ocultos no obxeto ou clase que os abarca e que nos provee de atributos e métodos faciles de manexar
- Herencia: As clases poden compartir información (atributos) e comportamento (métodos) e facer xerarquias
- Polimorfismo: Os obxetos poden ter diferente comportamento según o contexto e tamén poden ser tratados igual ainda que os tipos sexan distintos

# Algunhas consecuencias de usar POO

## As características básicas da POO

- Encapsulación: datos e operacións relacionados van xuntos e son accedidos soamente pola clase
- Abstracción: os conceptos complexos quedan ocultos no obxeto ou clase que os abarca e que nos provee de atributos e métodos faciles de manexar
- Herencia: As clases poden compartir información (atributos) e comportamento (métodos) e facer xerarquias
- Polimorfismo: Os obxetos poden ter diferente comportamento según o contexto e tamén poden ser tratados igual ainda que os tipos sexan distintos

# Algunhas consecuencias de usar POO

## As características básicas da POO

- Encapsulación: datos e operacións relacionados van xuntos e son accedidos soamente pola clase
- Abstracción: os conceptos complexos quedan ocultos no obxeto ou clase que os abarca e que nos provee de atributos e métodos faciles de manexar
- Herencia: As clases poden compartir información (atributos) e comportamento (métodos) e facer xerarquias
- Polimorfismo: Os obxetos poden ter diferente comportamento según o contexto e tamén poden ser tratados igual ainda que os tipos sexan distintos

# Outline



# Exemplo de obxeto: Clase Person

## O único sitio onde as persoas deben ser tratadas coma obxetos

```
import datetime as dt
class Person():
    DAYS_PER_YEAR = 365

    def __init__(self, name, birth_dt):
        self.name = name
        self.birth_dt = birth_dt
    def age(self):
        interval = dt.datetime.now() - self.birth_dt
        return interval.days // Person.DAYS_PER_YEAR

p = Person('Alex', dt.datetime(2003, 5, 13))

print(f'{p.name} ten {p.age()} anos ')
# Alex ten 20 anos
```

# Outline



# Motivación dos patróns

- A programación é tradicionalmente unha disciplina onde os profesionais van desde o artesanal ao enxeñeril
- Ou pra definilo en palabras dun home moi sabio.
- A programación é...

# Motivación dos patróns

- A programación é tradicionalmente unha disciplina onde os profesionais van desde o artesanal ao enxeñeril
- Ou pra definilo en palabras dun home moi sabio.
- A programación é...

# Motivación dos patróns

- A programación é tradicionalmente unha disciplina onde os profesionais van desde o artesanal ao enxeñeril
- Ou pra definilo en palabras dun home moi sabio.
- A programación é...

# Motivación dos patróns

## Aquí tamén aplica

"ORDEN & TALENTO"

Figure:

# Outline



# Patróns de deseño

## Qué me estás contando, neno?

- Un patrón de deseño é sencillamente a definición dunha solución eficaz e probada que pode ser reproducida e reutilizada en contextos semellantes
- O patrón consiste nunha descripción de como aplicar a solución de forma xeral
- Non se reduce meramente a reutilizar un método ou unha clase de utilidade específica, porque un patrón non é código
- O patrón ten unha natureza máis xeral. Non é como reutilizar código, que é moito máis específico

# Patróns de deseño

## Qué me estás contando, neno?

- Un patrón de deseño é sencillamente a definición dunha solución eficaz e probada que pode ser reproducida e reutilizada en contextos semellantes
- O patrón consiste nunha descripción de como aplicar a solución de forma xeral
- Non se reduce meramente a reutilizar un método ou unha clase de utilidade específica, porque un patrón non é código
- O patrón ten unha natureza máis xeral. Non é como reutilizar código, que é moito máis específico

# Patróns de deseño

## Qué me estás contando, neno?

- Un patrón de deseño é sencillamente a definición dunha solución eficaz e probada que pode ser reproducida e reutilizada en contextos semellantes
- O patrón consiste nunha descripción de como aplicar a solución de forma xeral
- Non se reduce meramente a reutilizar un método ou unha clase de utilidade específica, porque un patrón non é código
- O patrón ten unha natureza máis xeral. Non é como reutilizar código, que é moito máis específico

# Patróns de deseño

## Qué me estás contando, neno?

- Un patrón de deseño é sencillamente a definición dunha solución eficaz e probada que pode ser reproducida e reutilizada en contextos semellantes
- O patrón consiste nunha descripción de como aplicar a solución de forma xeral
- Non se reduce meramente a reutilizar un método ou unha clase de utilidade específica, porque un patrón non é código
- O patrón ten unha natureza máis xeral. Non é como reutilizar código, que é moito máis específico

# Motivacións dos patróns

## Vantaxes

- Aforrannos reinventar a roda
- Están probados e testeados
- Código máis lexible
- Nos dá un vocabulario común pra entendernos

# Motivacións dos patróns

## Vantaxes

- Aforrannos reinventar a roda
- Están probados e testeados
- Código máis lexible
- Nos dá un vocabulario común pra entendernos

# Motivacións dos patróns

## Vantaxes

- Aforrannos reinventar a roda
- Están probados e testeados
- Código máis lexible
- Nos dá un vocabulario común pra entendernos

# Motivacións dos patróns

## Vantaxes

- Aforrannos reinventar a roda
- Están probados e testeados
- Código máis lexible
- Nos dá un vocabulario común pra entendernos

# Outline



# O noso patrón

- Hai moitos patróns de deseño e de varios tipos distintos según a sua función
- Nós hoxe imos presentar un patrón sinxelo
- Pra todos vós, aquí temos o patrón... (redoble de tambores)

# O noso patrón

- Hai moitos patróns de deseño e de varios tipos distintos según a sua función
- Nós hoxe imos presentar un patrón sinxelo
- Pra todos vós, aquí temos o patrón... (redoble de tambores)

# O noso patrón

- Hai moitos patróns de deseño e de varios tipos distintos según a sua función
- Nós hoxe imos presentar un patrón sinxelo
- Pra todos vós, aquí temos o patrón... (redoble de tambores)

# O noso patrón

OBSERVADOR

Figure:

# Definición e descripción

## Como se define

| | |
|---|---|
| Nome | Observador |
| Problema | En moitas partes precisamos comunicar información con moitos obxetos distintos |
| Solución | Un sistema de suscripción polo cal os obxetos observadores son notificados polo observado |
| Consecuencia | O acoplamento é moito máis lixeiro, e como os obxetos apenas se coñecen, é difícil que un cambio nun afecte aos demáis |

# Definición e descripción

## Estructura

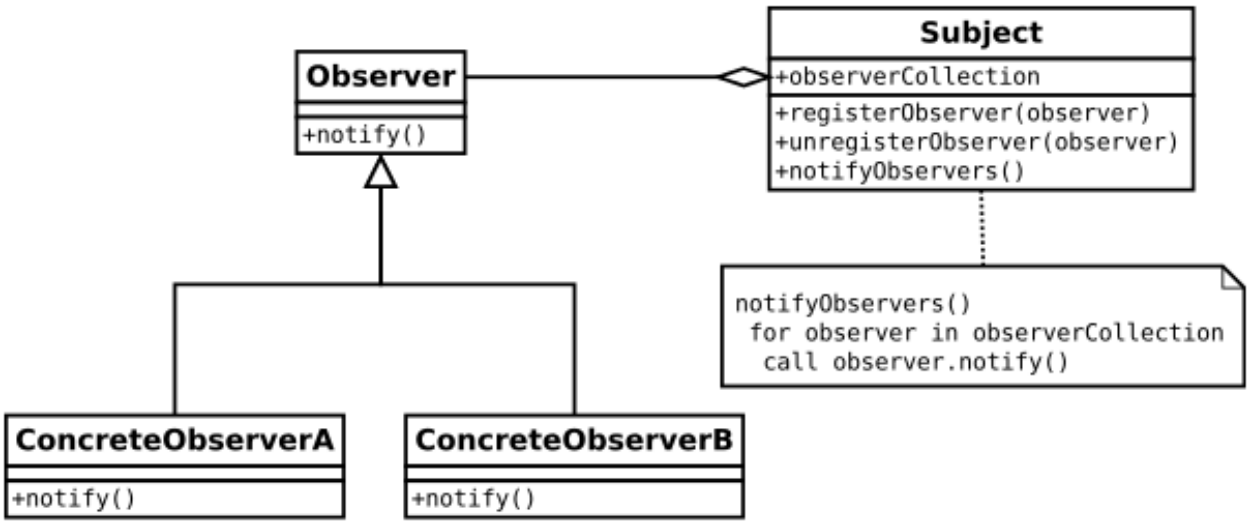


Figure:

# Implementación

```
1  class Observable:
2      def __init__(self):
3          self._observers = []
4
5      def register_observer(self, observer):
6          self._observers.append(observer)
7
8      def notify_observers(self, *args, **kwargs):
9          for obs in self._observers:
10             obs.notify(self, *args, **kwargs)
11
12 class Observer:
13     def __init__(self, observable):
14         observable.register_observer(self)
15
16     def notify(self, observable, *args, **kwargs):
17         print("Notificando ", args, kwargs, "desde", observal
```

# Resumo

- Hai infinidade de patróns pra resolver problemas. Pode ser que ti implementases algún sen decatarte
- Os patróns nos dan recursos pra facer frente a problemas con solucións probadas, efectivas e documentadas

# Resumo

- Hai infinidade de patróns pra resolver problemas. Pode ser que ti implementases algún sen decatarte
- Os patróns nos dan recursos pra facer frente a problemas con solucións probadas, efectivas e documentadas

# Pra ler máis I

- https://en.wikibooks.org/wiki/Computer_Science_Design_Patterns
- https://es.wikipedia.org/wiki/Categor%C3%ADa:Patrones_de_dise%C3%B1o
- Head First Design Patterns (O'Reilly)
- Python: Master the Art of Design Patterns