python

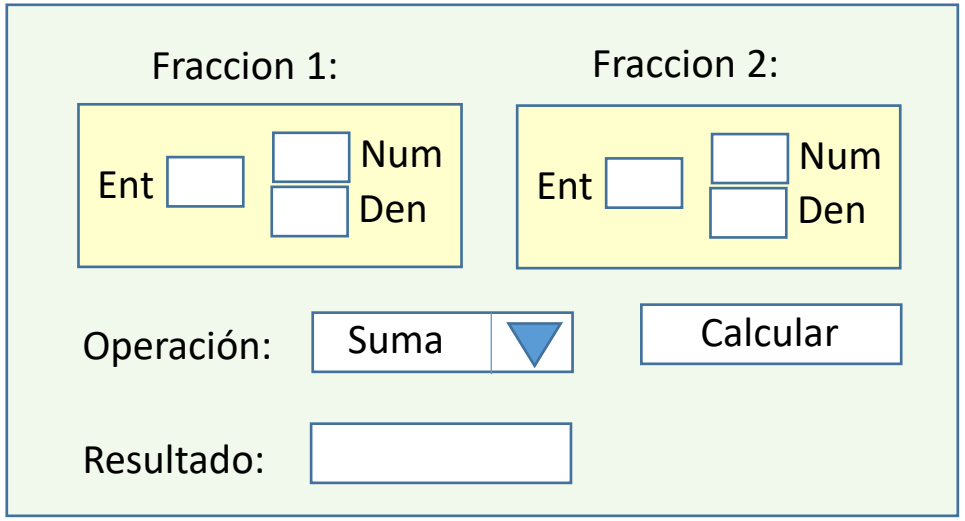
Aplicaciones de Escritorio con Tkinter

Turbo Código

04

Ejemplo práctico

Contenido



- Ejemplo práctico:
  - Operaciones con fracciones mixtas

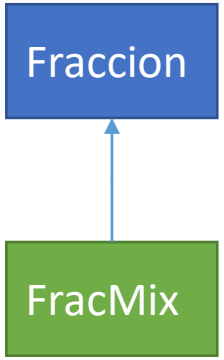
- Funciones dentro de funciones
- Herencia: Reutilizando código de la clase padre
- Diseño de pantalla

Funciones dentro de funciones

- Puedes declarar una función dentro de otra función
- La vida de la función interna es efímera al pertenecer al ámbito de la función donde es definida.
- La función interna va a ser creada de nuevo cada vez que llames a la función donde es creada y dejará de existir cuando esta termine. Esto implica que no podrá ser llamada desde fuera de la función donde es definida.

```
def simplifica(self):
    d=self.mcd(self.num,self.den)
    self.num=int(self.num/d)
    self.den=int(self.den/d)

def mcd(self,a,b):
    if b==0:
      return a
    else:
      return self.mcd(b,a%b)
```

```
def simplifica(self):

    def mcd(a,b):
      if b==0:
        return a
      else:
        return mcd(b,a%b)

    d=mcd(self.num,self.den)
    self.num=int(self.num/d)
    self.den=int(self.den/d)
```

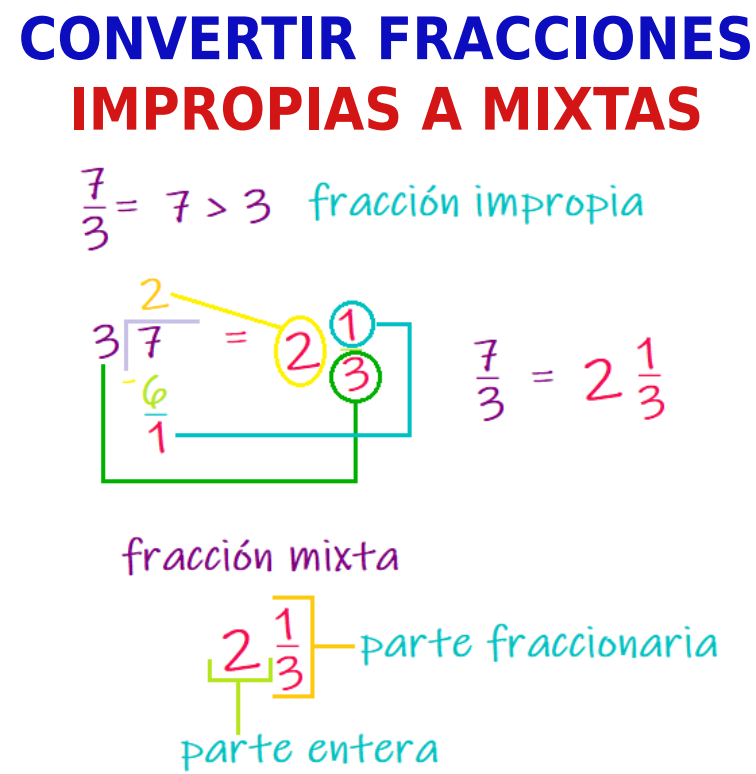


Herencia

(Reutilizando código de la clase padre)

```
class Fraccion:

    def __init__(self, num=0, den=1):···

    def __str__(self):···

    def __mul__(self, obj):···

    def __div__(self, obj):···

    def __add__(self, obj):···

    def __sub__(self, obj):···

    def __eq__(self, b):···

    def simplifica(self):···
```

```
class FracMix(Fraccion):

    def __init__(self,ent,num=0,den=1):
        self.ent=ent
        super().__init__(num,den)
        self.simplifica()
        super().simplifica()

    def __str__(self):
        return str(self.ent) + super().__str__()

    def simplifica(self):
        if self.num > self.den:
            aux=self.num//self.den
            self.ent=self.ent+aux
            self.num-=(aux*self.den)
```

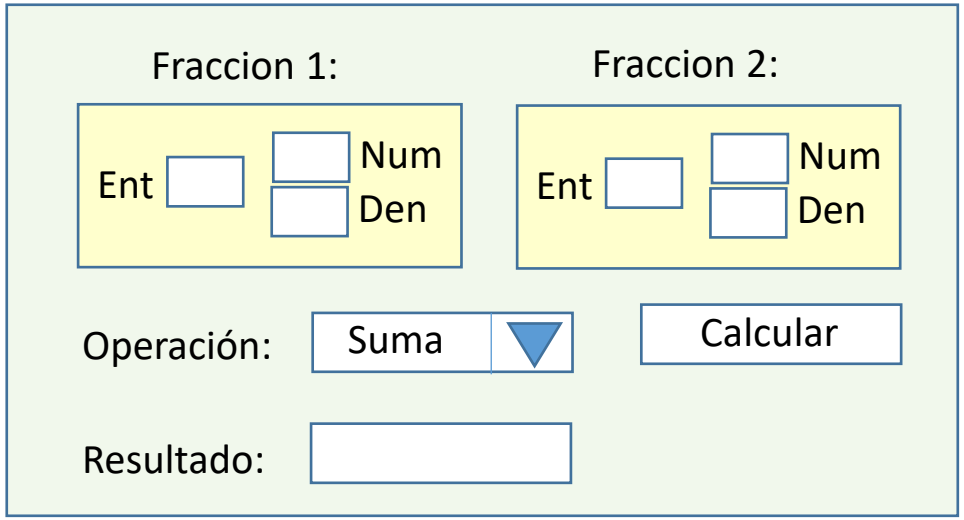


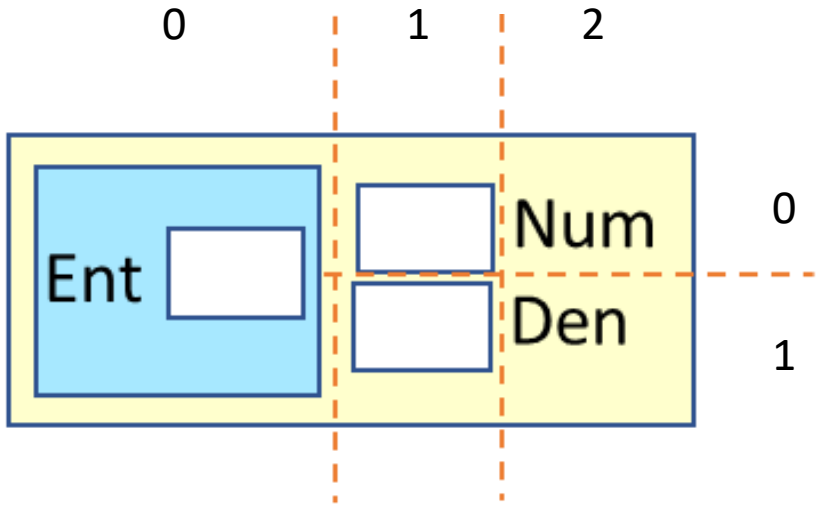
Turbo Código

https://www.youtube.com/user/juarezefren

Operator	Method	Expression
+ Addition	`__add__(self, other)`	`a1 + a2`
- Subtraction	`__sub__(self, other)`	`a1 - a2`
* Multiplication	`__mul__(self, other)`	`a1 * a2`
@ Matrix Multiplication	`__matmul__(self, other)`	`a1 @ a2` (*Python 3.5*)
/ Division	`__div__(self, other)`	`a1 / a2` (*Python 2 only*)
/ Division	`__truediv__(self, other)`	`a1 / a2` (*Python 3*)
// Floor Division	`__floordiv__(self, other)`	`a1 // a2`
% Modulo/Remainder	`__mod__(self, other)`	`a1 % a2`
** Power	`__pow__(self, other[, modulo])`	`a1 ** a2`
<< Bitwise Left Shift	`__lshift__(self, other)`	`a1 << a2`
>> Bitwise Right Shift	`__rshift__(self, other)`	`a1 >> a2`
& Bitwise AND	`__and__(self, other)`	`a1 & a2`
^ Bitwise XOR	`__xor__(self, other)`	`a1 ^ a2`
| (Bitwise OR)	`__or__(self, other)`	`a1 | a2`
- Negation (Arithmetic)	`__neg__(self)`	`-a1`
+ Positive	`__pos__(self)`	`+a1`
~ Bitwise NOT	`__invert__(self)`	`~a1`
< Less than	`__lt__(self, other)`	`a1 < a2`
<= Less than or Equal to	`__le__(self, other)`	`a1 <= a2`
== Equal to	`__eq__(self, other)`	`a1 == a2`
!= Not Equal to	`__ne__(self, other)`	`a1 != a2`
> Greater than	`__gt__(self, other)`	`a1 > a2`
>= Greater than or Equal to	`__ge__(self, other)`	`a1 >= a2`
[index] Index operator	`__getitem__(self, index)`	`a1[index]`
in In operator	`__contains__(self, other)`	`a2 in a1`
(*args, ...) Calling	`__call__(self, *args, **kwargs)`	`a1(*args, **kwargs)`

Diseño de pantalla





python

Aplicaciones de Escritorio con Tkinter

Turbo Código

04

Ejemplo práctico