3. Llegando al final

- Llegando al final
- Programa 11
- Programa 12
- Programa 13
- Programa 14
- Ejercicios de autoevaluación
- Soluciones a los ejercicios de autoevaluación

Copyright 2026

Llegando al final

Objetivos

- Aprender a manejarse con tipos de variable avanzados.
- Aprender a utilizar funciones.

Copyright 2026

Programa 11

Descripción del problema:

Se desea realizar un programa que guarde el nombre, la edad y lo que se les recomendaba hacer, como teníamos antes, para un número de presentes que el usuario dirá. Después de preguntar la información de cada uno de ellos, el programa debe preguntar un número y dar la información referida a la persona que se introdujo en esa posición.

Materia nueva:

Los Diccionarios son un tipo de datos que puede guardar para una sola variable un conjunto de campos. Pongamos un ejemplo: Imaginemos un alumno. Existe un montón de información que podríamos querer guardar sobre él y hacer una tabla como:

|Nombre|Apellido 1|Apellido 2|Altura|Nº Calzado|Color de pelo| |--|--|--|--|--|--| |Jorge|Bes|Tuán|2 m|47|Negro|

Cuando alguien nos preguntara por este alumno, lo haría diciéndonos: Dime el primer apellido del alumno, el color de pelo... Es decir, nos diría el campo que quiere. Tras esto, nosotros iríamos a la celda y responderíamos con la información allí contenida.

Bueno, pues esto es un Diccionario para Python. Tiene unos valores que se llaman Key que en este caso sería la fila superior (Nombre, Apellido1...) y otros que se llaman Value que serían los de la fila inferior.

Una duda que suele asaltar es: ¿Por qué sólo una fila? ¿Y si queremos guardar un conjunto de 1000 alumnos? Pues muy sencillo, se hace una lista de diccionarios, que es lo que vamos a hacer en nuestro programa. Al fin y al cabo, una lista es un conjunto de cosas, nadie ha dicho que haya algo que no se pueda meter ahí.

La sintaxis para declarar un diccionario vacío es:

diccionario **={}**

Y para guardar cualquier dato es:

diccionario[“Key1”]=datoaguardar

Copyright 2026

Aquí hay que decir que, a diferencia de las listas, donde había que añadir un nuevo componente con una orden, aquí es tan fácil como poner una nueva "Key" para que la añada al conjunto de ellas. Es decir, es como tener una tabla dinámica donde podemos añadir columnas sobre la marcha.

Un diccionario tiene muchísimas más posibilidades en Python, como se puede ver en:

https://docs.python.org/3/library/stdtypes.html#typesmapping

No obstante, en este curso no se va sino a introducir este tipo de variable. Queda demostrada la potencia de Python y el nivel de abstracción de los datos y se anima a los alumnos a completar su formación conforme lo vayan necesitando para sus proyectos.

NOTA: Si vas a introducir cambios en un diccionario dentro de una estructura como **for** o **while**, que los ejecutará varias veces, declara esa variable dentro de la estructura y no fuera. Más adelante se explica el por qué.

Solución

Algoritmo:

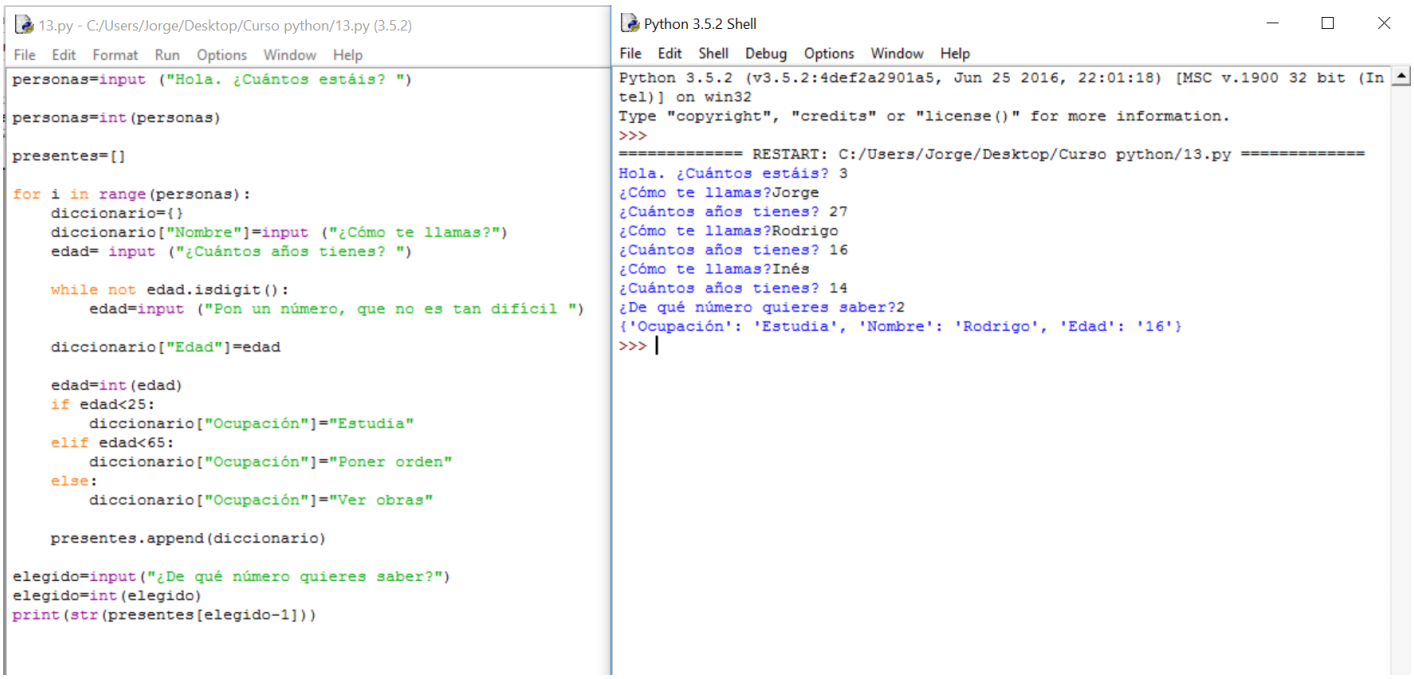
1.- Preguntaremos cuánta gente hay

2.- A cada uno le preguntaremos su nombre y su edad. Guardaremos las dos variables anteriores y la tarea que tienen que hacer (poner orden, estudiar y mirar obras) en un diccionario.

3.- Pediremos que nos diga el número de persona que quiere que le mostremos.

4.- Lo mostraremos en pantalla.

Solución:



Comentarios:

Aquí hay que comentar un error bastante curioso que tiene más que ver con cómo hace las cosas Python que con la programación en sí. Prueba a cambiar la declaración diccionario={} a justo antes del **for** y usa la siguiente orden al final del programa:

print (str(presentes**))** Para que saque por pantalla toda la lista que hemos introducido.

Verás que ha guardado el último tantas veces como gente había. Esto es porque, cada vez que se declara, hace uno nuevo, si no, sólo se está alterando el mismo todo el rato.

No hace falta que lo entiendas, realmente es un error de Python, aunque conozco que se da en varios lenguajes más. Simplemente, quédate con que las variables, para que se puedan usar varias veces, es necesario que tengan su espacio en memoria diferente, y eso lo haces declarándolas de nuevo ya que Python les busca un nuevo hueco en memoria aunque las llame igual.

Es necesario también destacar el recurso de restar 1 a la variable que nos dice qué persona queremos para ajustar el que el ordenador empiece a contar por el 0 y nosotros por el 1.

Copyright 2026

Programa 12

Descripción del problema:

Si miramos el programa número 11, nos damos cuenta de que es todo como un spaghetti, empieza en la primera orden y es todo secuencial hasta el final. Además, a la hora de la lectura, eso de ver el trozo donde se mira la ocupación con tres declaraciones de dicha parte del diccionario no la facilita, como se ve a continuación.

```
edad=int(edad)
if edad<25:
    diccionario["Ocupación"]="Estudia"
elif edad<65:
    diccionario["Ocupación"]="Poner orden"
else:
    diccionario["Ocupación"]="Ver obras"
```

Este tipo de programación se le llama Programación Spaghetti. El programa que hemos hecho es sumamente simple pero, si imaginamos un programa de 2000 líneas, el problema de lectura puede ser tan serio que retocar algo del mismo nos suponga un dolor.

Para evitar esto existen las funciones, que no es otra cosa que sacar código del hilo principal para llamarlo cuando nos convenga. De esta forma, tendremos el **programa** muy claro para leer y retocar.

El programa 12 consiste en extraer la parte de la ocupación del hilo principal.

Materia nueva:

Una función se define en Python de la siguiente manera:

```
def nombredelafuncion (variablesquevaausar)
Ordenes
return(variableadevolver)
```

Copyright 2026

Para recibir el dato que devuelve (**return** significa devolver) es necesario guardarlo en algún sitio. Lo habitual es asignarlo a una variable:

```
variable=nombredelafuncion (variablequeleentregamos)
```

Respecto a la ubicación de las funciones, lo habitual es declararlas antes de que empiece el programa. Por lo tanto, las primeras líneas se dedican a escribir funciones y el hilo principal del programa viene después.

Una vez definida, la llamada es tan simple como poner el nombre de la función y decirle qué variables va a usar.

¡Cuidado! Las variables que va a usar no son aquellas con las que se le llama. Digamos que la función recibe unas variables pero, para no modificar las del programa, hace una copia cambiándoles el nombre y usa las que ha copiado.

NOTA: Si te has dado cuenta, a lo largo de todo el curso he hablado de "órdenes", pues bien, era mentira: son "funciones". Se ha hecho así porque era demasiado duro empezar a hablar de algo que quedaba muy lejos en el progreso de aprendizaje; mejor ver primero para qué sirven y luego ya se explicará el por qué.

Solución

Algoritmo:

1.- Preguntaremos cuánta gente hay

2.- Para cada uno de ellos se les pregunta y:

- 2.1.- En un diccionario iremos guardando nombre y edad.
- 2.2.- A la hora de elegir ocupación haremos una llamada a una función que lo calculará *Problema

3.- Pediremos que nos diga el número de persona que quiere que le mostremos.

4.- Lo mostraremos en pantalla.

Solución:

14.py - C:/Users/Jorge/Desktop/Curso python/14.py (3.5.2)

File Edit Format Run Options Window Help

```
def obtenerocupacion (numero):

    if numero<25:
        curro="Estudia"
    elif numero<65:
        curro="Poner orden"
    else:
        curro="Ver obras"
    return curro

personas=input ("Hola. ¿Cuántos estáis? ")

personas=int(personas)

presentes=[]

for i in range(personas):
    diccionario={}
    diccionario["Nombre"]=input ("¿Cómo te llamas?")
    edad= input ("¿Cuántos años tienes? ")

    while not edad.isdigit():
        edad=input ("Pon un número, que no es tan difícil ")

    diccionario["Edad"]=edad

    edad=int(edad)
    diccionario["Ocupación"]=obtenerocupacion(edad)
    presentes.append(diccionario)

elegido=input("¿De qué número quieres saber?")
elegido=int(elegido)
print(str(presentes[elegido-1]))
```

Python 3.5.2 Shell

File Edit Shell Debug Options Window Help

```
Python 3.5.2 (v3.5.2:4def2a2901a5, Jun 25 2016, 22:01:18) [MSC v.1900 32 bit (In
tel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: C:/Users/Jorge/Desktop/Curso python/14.py ==========
Hola. ¿Cuántos estáis? 3
¿Cómo te llamas?Jorge
¿Cuántos años tienes? 27
¿Cómo te llamas?Rodrigo
¿Cuántos años tienes? 16
¿Cómo te llamas?Inés
¿Cuántos años tienes? 14
¿De qué número quieres saber?3
{'Edad': '14', 'Nombre': 'Inés', 'Ocupación': 'Estudia'}
>>>
```

Copyright 2026

Programa 13

Descripción del problema:

Se desea realizar un programa que devuelva el máximo común divisor de dos números. Se debe realizar por medio de funciones.

Materia nueva:

Una función puede devolver tan sólo un único valor. La solución es devolver una lista que, a fin y al cabo, es una única variable.

En este programa, devolveremos los divisores de cada número en una lista.

Solución

Algoritmo:

1.- Pedimos los dos números

2.- Calculamos sus divisores

3.- El mcd se obtiene comparando las dos listas. El mayor divisor que esté en las dos listas será el máximo común divisor. La forma más rápida es:

3.1.- Comparar el más pequeño del primer número con todos los del otro número. Lo guardo si coincide.

3.2.- Comparo el siguiente del primer número con todos los del otro número. Lo guardo si coincide.

3.3.- Cuando haya terminado, el último número que tenga guardado será el mcd.

Solución:

Copyright 2026

15.py - C:/Users/Jorge/Desktop/Curso python/15.py (3.5.2)

File Edit Format Run Options Window Help

```
def obtenerdivisores (numero):
    divisores=[]
    for i in range(1,numero+1):
        if numero%i==0:
            divisores.append(i)
    return divisores

numero1=int(input("Indroduce el primer número: "))
numero2=int(input("Introduce el segundo número: "))
divisores1=obtenerdivisores(numero1)
divisores2=obtenerdivisores(numero2)
for i in divisores1:
    for z in divisores2:
        if i==z:
            mcd=i
            encontrado=True

print ("El máximo común divisor es: "+str(mcd))
```

Python 3.5.2 Shell

File Edit Shell Debug Options Window Help

```
Python 3.5.2 (v3.5.2:4def2a2901a5, Jun 25 2016, 22:01:18) [MSC v.1900 32 bit
tel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: C:/Users/Jorge/Desktop/Curso python/15.py ==========
Indroduce el primer número: 81
Introduce el segundo número: 45
El máximo común divisor es: 9
>>>
```

Comentarios:

El mayor problema de este programa es realizar un algoritmo que calcule el mínimo común divisor. Si las dificultades vienen de ahí, no pasa nada, se irá alcanzando la suficiencia algorítmica con el tiempo y la práctica.

Copyright 2026

Programa 14

Descripción del problema:

Se desea realizar un programa en el que el usuario vaya introduciendo números hasta que introduzca algo que no lo sea. Una vez introducidos, se añadirá a cada número la información de si es primo o no, el número de cifras que tiene y si es el mínimo o el máximo. Al final, sacará todo por pantalla. Es necesario usar un diccionario, listas y funciones.

Materia nueva:

En este caso, nos falta por saber cómo conocer la longitud de una cadena (es una de las múltiples funciones que nos quedan en el tintero). Dicha función es:

len(string**)** Aquí hay una curiosidad. Si la mayor parte eran string**.función**, ¿por qué no hay ningún punto? Pues sencillamente porque esta función se puede usar para un montón de cosas aparte de una Cadena de caracteres. De momento la usaremos para conocer cuántos dígitos tiene el número que el usuario ha introducido.

NOTA:

- **Modo avanzado:** Recuerda que, para cada variable nueva tipo diccionario, es necesario declararla para que no apunte a la misma dirección de memoria.
- **Modo terrestre:** Mete la declaración en el bucle para que se declare y sea nueva cada vez que se usa.

Solución

Algoritmo:

1.- Pedir números hasta que haya una introducción que no se pueda convertir en uno. Ir guardándolos en una lista

2.- Para cada uno de ellos:

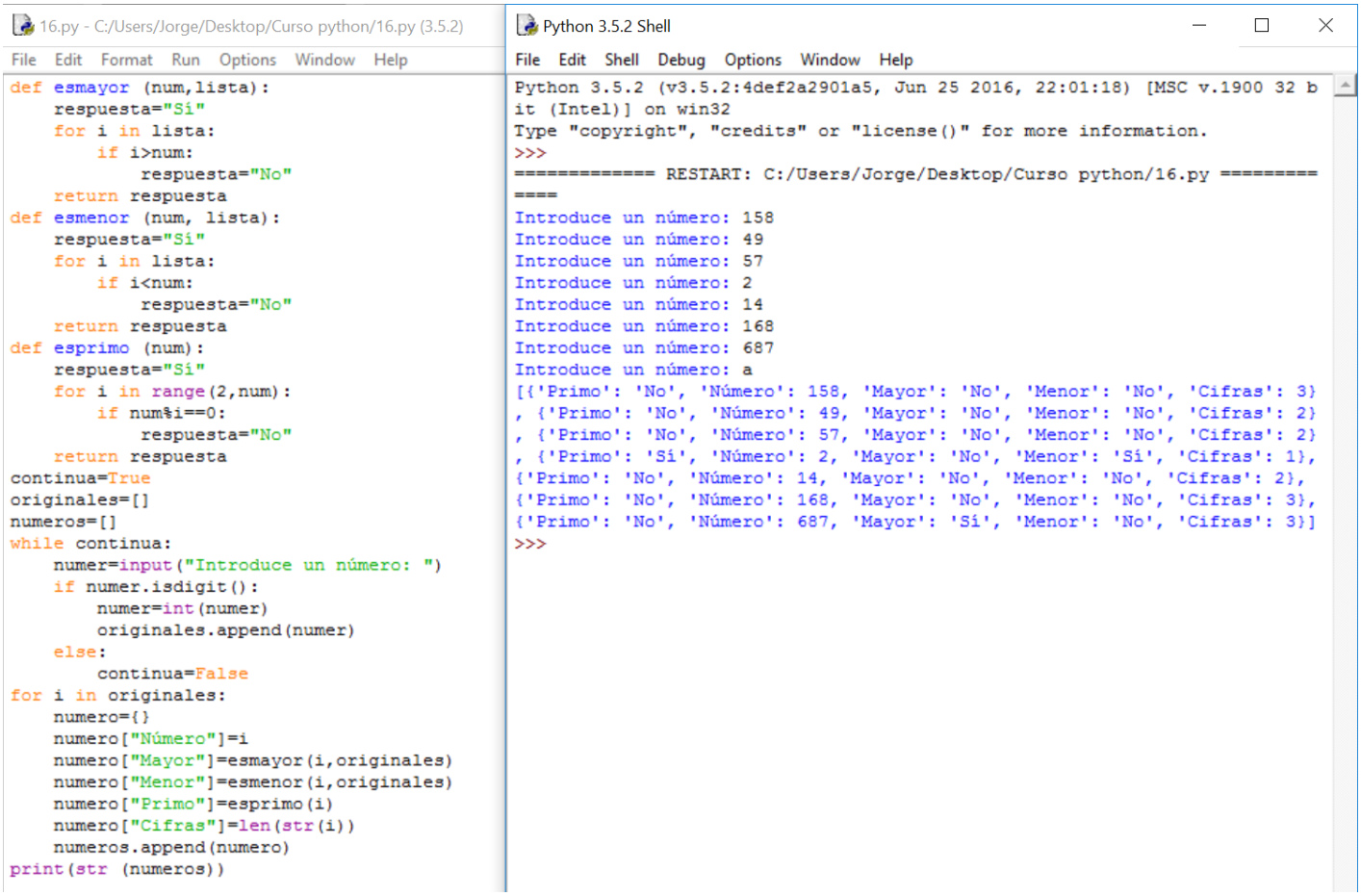
- 2.1.- Guardarlo bajo la "Key": "Número"

Copyright 2026

- 2.2.- Comprobar si es menor, mayor y primo y guardarlo bajo las "Keys": "Menor", "Mayor" y "Primo"
- 2.3.- Contar con la función **len(** s **)** la cantidad de caracteres del número pasado a Cadena de caracteres y guardarlo bajo la "Key": "Cifras"

3.- Sacar la lista de diccionarios por la pantalla.

Solución:



Explicación:

En este caso, en las órdenes:

numero["Mayor"]=esmayor(i,originales) y esmenor(i,originales) podemos ver que se ha trasladado a la función una lista entera.

Podría parecer que sólo admitiría variables simples pero, realmente, no existe límite de datos o variables a la hora de pasar a las funciones.

Sin embargo, cada función usa una lista diferente. Sí, en Python no se permite usar el original de ninguna variable enviada a una función. Si nos pusiéramos puristas, esto duplicaría la cantidad de memoria reservada en el ordenador a nuestro programa; es verdad, pero la cantidad de memoria y la capacidad de cálculo han dejado de ser un problema hace mucho tiempo.

Respecto a la función **len(**s**)**, ha habido que pasarlo primero a Cadena de caracteres para poder realizarlo.

Copyright 2026

Ejercicios de autoevaluación

Felicidades por haber finalizado el Módulo 3. A continuación se muestran los programas que servirán de repaso a lo aprendido:

1.- Realiza un programa que calcule el mínimo común múltiplo de dos números.

Como ayuda: Habrá que realizar una lista con los múltiplos de cada número hasta el producto de los dos que el usuario no dé, ya que será el máximo valor del mcm.

Por lo demás, el proceso es igual que en el mcd pero teniendo en cuenta que, si empezamos buscando desde el valor más bajo, una vez que encuentra la primera coincidencia en las dos listas, ya no querremos que vuelva a cambiar ese valor. Hay una orden que se llama **break** y permite salir del bucle **for** o **while** donde está metido. En este caso no nos ayudará porque estamos metidos en dos. Es necesario usar una booleana que registre si ya ha sido localizado o no y, por tanto, permitir o no, el cambio ante más coincidencias.

Si se hace la lista de multiplicadores empezando por el más alto, nos ahorramos ese pequeño tinglado.

2.- Realiza un programa que, dada una fracción, obtenga la fracción irreducible (la más simplificada posible). El usuario introducirá el numerador y denominador y la salida será un diccionario con "Numerador" y "Denominador" como Keys.****

3.- Realiza un programa que, dadas dos fracciones, obtenga las fracciones equivalentes que tengan el mismo denominador siendo éste el mínimo posible.

4.- Realiza una calculadora de fracciones. Debe poder sumar, restar, multiplicar y dividir dos fracciones. El resultado debe ser una fracción irreducible.

En este caso, como el resultado puede ser negativo, debemos usar una función matemática que nos dé el valor absoluto de un número ya que, si usamos los bucles que hemos realizado hasta ahora, tendremos problemas. Dicha función es **abs(**número**)**

En la solución se han cambiado ciertas funciones que había en el resto de programas para más eficiencia. No es preocupante si os sale con más líneas.

NOTA: Estos 4 programas simbolizan la máxima de la programación: Dividir en problemas más pequeños. Aplícalo siempre en tus programas porque, aunque en este curso no ha habido grandes

Copyright 2026

complicaciones, según se quieran resolver problemas más complejos, no es que sea una máxima sino el único camino.

Mi más cordial enhorabuena por haber terminado el curso. Espero que te sea útil y haya sido el primer paso hacia la maestría en programación.

Copyright 2026

Soluciones a los ejercicios de autoevaluación

Autoevaluacion 3.1.py - C:/Users/Jorge/Desktop/Curso python/Autoevaluaci

File Edit Format Run Options Window Help

```
def obtenermultiplos (numero,elotro):
    multiplos=[]
    for i in range(elotro,0,-1):
            multiplos.append(i*numero)
    return multiplos

numero1=int(input("Indroduce el primer número: "))
numero2=int(input("Introduce el segundo número: "))
multiplos1=obtenermultiplos(numero1,numero2)
multiplos2=obtenermultiplos(numero2,numero1)
for i in multiplos1:
    for z in multiplos2:
        if i==z:
            mcm=i
            encontrado=True

print ("El mínimo común múltiplo es: "+str(mcm))
```

Python 3.5.2 Shell

File Edit Shell Debug Options Window Help

```
Python 3.5.2 (v3.5.2:4def2a2901a5, Jun 25 2016, 22:01:18) [MSC v.1900 32 bit (In
tel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:/Users/Jorge/Desktop/Curso python/Autoevaluacion 3.1.py =====
Indroduce el primer número: 12
Introduce el segundo número: 18
El mínimo común múltiplo es: 36
>>>
```

Autoevaluacion 3.2.py - C:\Users\Jorge\Desktop\Curso python\Autoeval

File Edit Format Run Options Window Help

```
def obtenerdivisores (numero):
    divisores=[]
    for i in range(1,numero+1):
        if numero%i==0:
            divisores.append(i)
    return divisores

numero1=int(input("Indroduce el numerador: "))
numero2=int(input("Introduce el denominador: "))
divisores1=obtenerdivisores(numero1)
divisores2=obtenerdivisores(numero2)
for i in divisores1:
    for z in divisores2:
        if i==z:
            mcd=i
            encontrado=True

fraccion={}
fraccion["Numerador"]=numero1/mcd
fraccion["Denominador"]=numero2/mcd
print (str (fraccion))
```

Python 3.5.2 Shell

File Edit Shell Debug Options Window Help

```
Python 3.5.2 (v3.5.2:4def2a2901a5, Jun 25 2016, 22:01:18) [MSC v.1900 32 bit (In
tel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\Jorge\Desktop\Curso python\Autoevaluacion 3.2.py =====
Indroduce el numerador: 95
Introduce el denominador: 245
{'Numerador': 19.0, 'Denominador': 49.0}
>>>
```

Autoevaluacion 3.3.py - C:\Users\Jorge\Desktop\Curso python\Autoevaluacion 3.3.py (3.5.2)

File Edit Format Run Options Window Help

```
def obtenerdivisores (numero1,numero2):
    divisores=[]
    for i in range(1,numero1+1):
        if numero1%i==0 and numero2%i==0:
            divisores.append(i)
    return divisores

fraccion1={}
fraccion2={}
fraccion1["Numerador"]=int(input("Indroduce el numerador1: "))
fraccion1["Denominador"]=int(input("Introduce el denominador1: "))
fraccion2["Numerador"]=int(input("Indroduce el numerador2: "))
fraccion2["Denominador"]=int(input("Introduce el denominador2: "))
""" Se va a conseguir denominador común multiplicando el denominador
de una fracción por la otra y viceversa"""

fraccion1["Numerador"]=fraccion1["Numerador"]*fraccion2["Denominador"]
fraccion2["Numerador"]=fraccion2["Numerador"]*fraccion1["Denominador"]
fraccion1["Denominador"]=fraccion1["Denominador"]*fraccion2["Denominador"]
fraccion2["Denominador"]=fraccion1["Denominador"]
""" Ahora dividiré por el mayor divisor común a ambas fracciones"""
print (str (fraccion1)+ "\n"+str(fraccion2))
divisores1=obtenerdivisores(fraccion1["Numerador"],fraccion1["Denominador"])
divisores2=obtenerdivisores(fraccion2["Numerador"],fraccion2["Denominador"])

for i in divisores1:
    for z in divisores2:
        if i==z:
            mcd=i

fraccion1["Numerador"]=fraccion1["Numerador"]/mcd
fraccion1["Denominador"]=fraccion1["Denominador"]/mcd
fraccion2["Numerador"]=fraccion2["Numerador"]/mcd
fraccion2["Denominador"]=fraccion1["Denominador"]
print (str (fraccion1)+ "\n"+str(fraccion2))
```

Python 3.5.2 Shell

File Edit Shell Debug Options Window Help

```
Python 3.5.2 (v3.5.2:4def2a2901a5, Jun 25 2016, 22:01:18) [MSC v.1900 32 I
tel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\Jorge\Desktop\Curso python\Autoevaluacion 3.3.py
Indroduce el numerador1: 12
Introduce el denominador1: 18
Indroduce el numerador2: 72
Introduce el denominador2: 24
{'Numerador': 288, 'Denominador': 432}
{'Numerador': 1296, 'Denominador': 432}
{'Numerador': 2.0, 'Denominador': 3.0}
{'Numerador': 9.0, 'Denominador': 3.0}
>>>
```

En este caso, hay un paso intermedio que se muestra. Conforme los programas se complican, es necesario decirle que saque por pantalla ciertos valores en determinados momentos para controlar que se está haciendo lo que queremos. Imagina si hubiera 2000 líneas.

Los valores correctos son los últimos.

Autoevaluacion 3.4.py - C:/Users/Jorge/Desktop/Curso python/Autoevaluacion 3.4.py (3.5.2)

File Edit Format Run Options Window Help

```
def mcd (numero1,numero2):
    for i in range(1,numero1+1):
        if numero1%i==0 and numero2%i==0:
            maxcomdiv=i
    return maxcomdiv
def suma(fracc1,fracc2,opera):
    resul={}
    if opera=="+":
        resul["Numerador"]=fracc1["Numerador"]*fracc2["Denominador"]+fracc2["Numerador"]*fracc1["Denominador"]
    else:
        resul["Numerador"]=fracc1["Numerador"]*fracc2["Denominador"]-fracc2["Numerador"]*fracc1["Denominador"]
    resul["Denominador"]=fracc1["Denominador"]*fracc2["Denominador"]
    return resul

fraccion1={}
fraccion2={}
resultado={}
fraccion1["Numerador"]=int(input("Indroduce el numerador1: "))
fraccion1["Denominador"]=int(input("Introduce el denominador1: "))
operacion=input("Introduce la operación que deseas ")
fraccion2["Numerador"]=int(input("Indroduce el numerador2: "))
fraccion2["Denominador"]=int(input("Introduce el denominador2: "))
if operacion=="+" or operacion=="-":
    resultado=suma(fraccion1,fraccion2,operacion)
elif operacion=="*":
    resultado["Numerador"]=fraccion1["Numerador"]*fraccion2["Numerador"]
    resultado["Denominador"]=fraccion2["Denominador"]*fraccion1["Denominador"]
else:
    resultado["Numerador"]=fraccion1["Numerador"]*fraccion2["Denominador"]
    resultado["Denominador"]=fraccion1["Denominador"]*fraccion2["Numerador"]

mcdiv=mcd(abs(resultado["Numerador"]),abs(resultado["Denominador"]))
resultado["Numerador"]=resultado["Numerador"]/mcdiv
resultado["Denominador"]=resultado["Denominador"]/mcdiv

print (str (resultado))
```

Ln: 31 Col: 4

Python 3.5.2 Shell

File Edit Shell Debug Options Window Help

```
Python 3.5.2 (v3.5.2:4def2a2901a5, Jun 25 20
1)] on win32
Type "copyright", "credits" or "license()" f
>>>
===== RESTART: C:/Users/Jorge/Desktop/Curso
Indroduce el numerador1: 2
Introduce el denominador1: 3
Introduce la operación que deseas +
Indroduce el numerador2: 4
Introduce el denominador2: 5
{'Denominador': 15.0, 'Numerador': 22.0}
>>>
===== RESTART: C:/Users/Jorge/Desktop/Curso
Indroduce el numerador1: 2
Introduce el denominador1: 7
Introduce la operación que deseas -
Indroduce el numerador2: 14
Introduce el denominador2: 21
{'Denominador': 21.0, 'Numerador': -8.0}
>>>
===== RESTART: C:/Users/Jorge/Desktop/Curso
Indroduce el numerador1: 3
Introduce el denominador1: 5
Introduce la operación que deseas *
Indroduce el numerador2: 4
Introduce el denominador2: 10
{'Denominador': 25.0, 'Numerador': 6.0}
>>>
===== RESTART: C:/Users/Jorge/Desktop/Curso
Indroduce el numerador1: 2
Introduce el denominador1: 3
Introduce la operación que deseas /
Indroduce el numerador2: 1
Introduce el denominador2: 8
{'Numerador': 16.0, 'Denominador': 3.0}
>>>
```

En este caso se han optimizado ciertos algoritmos: Se ha utilizado la función **suma** para sumar y

restar y se ha cambiado la función para obtener el máximo común divisor. Esto es lo que suele pasar cuando se le dan varias vueltas al mismo programa. La solución óptima no es exigible a nuestro nivel pero irás viendo que cada vez eres capaz de obtener soluciones más eficientes.

Gracias por haber seguido el curso, te deseo muchos éxitos.

Copyright 2026