Area de un triángulo

El área de un triángulo es un poco chungo de calcular, porque depende del tipo de triángulo (equilátero, escaleno o isóceles).

Hay sin embargo una fórmula genérica. Para poder aplicarla, tenemos que calcular el *semi-perímetro* primero:

$$sp = \frac{(lado1 + lado2 + lado3)}{2}$$

Ahora el área se calcula como.

$$\sqrt{sp * (sp - lado1) * (sp - lado2) * (sp - lado3)}$$

Crea una función que:

- recibe los 3 lados
- usa la función `sqrt` que está en `math`
- devuelve el area

In [1]:

```
import math

def triangle_area(side1, side2, side3):
    """
    Calcula el area usando el semi-perímetro
    """
    sp = (side1 + side2 + side3) / 2

    return math.sqrt(sp * (sp - side1) * (sp - side2) * (sp - side3))

triangle_area(3,2,3)
```

Out[1]:

```
2.8284271247461903
```

Area de una esfera

La fórmula para el área de una esfera es la siguiente:

$$area = 4 * \pi * radio^2$$

1. Crea una función que calcula el area de la esfera
2. Usa el π de `math`

Volúmen de una esfera

$$volumen = \frac{4 * \pi * radio^3}{3}$$

2. Usa el π de `math`

Los gringos y sus unidades absurdas

Cuando vivía en EEUU, una de las cosas que más problemas me dió fueron sus unidades desquiciadas: onzas, onzas fluidas, millas, yardas y un sinfín de absurdeces que hacen que una tarea cotidiana se convierta en un desafío hercúleo.

La cosa, lamentablemente, no termina ahí. También usan subdivisiones de dichas unidades. Esas subdivisiones no pueden lógicamente, ser decimales. Normalmente usan octavos (tal cual).

Mi némesis particular: el octavo de milla

Vas por la carretera y waze te pega un grito, diciendo que tienes que salir a la derecha dentro de *3 octavos de milla*. ¿Mande?

Te pones a echar cuentas, y cuando terminas, ves que era justo ahí atrás.

Para evitar esto, vas a hacer una función que transforma octavos de milla en metros.

1. Recibes el número de octavos de milla
2. Tienes una variable dentro de tu fución con la correspondencia millas a metros (1609.344 metros por milla)
3. Con eso calculas el total de metros y lo devuelves

4. Usa la función `round` de Python (está en el ámbito global) para devolver un número entero (a nadie le

Edad

1. Crea una función que recibe tu año de nacimiento y calcula tu edad en años
2. Crea una función que hace lo mismo pero te devuelve la edad en días (olvídate de los años bisiestos)

¿Se podría reaprovechar al primera función a la hora de crear la segunda y no repetir código? **DRY**

BMI: índice de masa corporal

La operación bikini se aproxima y lo primero es saber cúanto tienes que adelgazar. Para eso, nada mejor que el BMI (https://www.nhlbi.nih.gov/health/educational/lose_wt/BMI/bmi-m.htm).

EL BMI es un índice (un número) que se obtiene a partir de tu peso y altura y según dicho valor, se sabe si están en tu peso ideal, por encima del mismo o por debajo (no te lo crees ni tú).

La fórmula, usando Kg para el peso y m para la altura es:

$$bmi = \frac{peso}{altura^2}$$

1. crea la función `bmi` que recibe peso y altura (en Kg y m)
2. devuelve el bmi

Euros a Pesetas

Una función para los nostálgicos: vamos a convertir precios en Euros a Pesetas. La tasa de conversión es: 166 Pesetas por Euro.

1. Crea la función `euro_to_pta` que recibe un número de euros y te devuelve la cantidad de Pesetas
2. Ahora crea la función `pta_to_euro` que hace lo opuesto.
    A. ¿Podrías implementar `pta_to_euro` llamando a `euro_to_pta` (para reaprovechar código)?

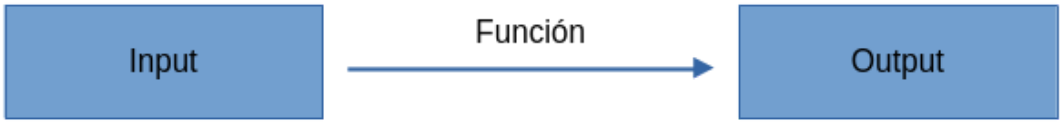
Energía cinética

Crea la función cinetic_energy(mass, speed) que acepta la masa y la velocidad de un objeto y devuelve la energía cinética del objeto, calculada con la fórmula: E = (1/2)mv².

👲 🧢 Funciones Reversibles

Las funciones son mecanismos que transforman una cosa en otra: te llevan de A a B. Por ejemplo, te llevan de (peso, altura) a bmi, o de radio a area, etc.

> Se dice que **mapean** su entrada a su salida



A veces, se puede conseguir encontrar la *función inversa* de una cierta función:

Si *f* es una función que te lleva de *A* a *B*,

entonces *f_inv* es una función que te lleva de *B* a *A*

> **f_inv** es la **inversa** de **f**

Ejercicio

1. La función ¿ `euro_to_peseta` es reversible?
2. La función `abs` de Python ¿es reversible?
3. La función `bmi` es reversible?

Funciones reversibles estimables

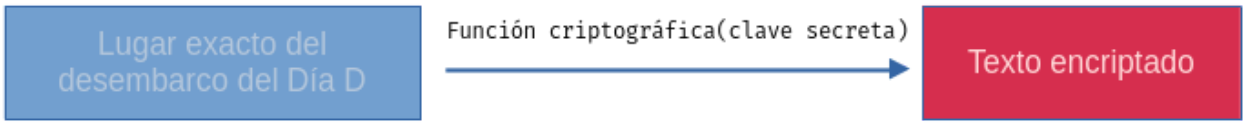
Son aquellas que, aunque no se puede obtnere una función inversa de forma directa, *si tenemos más información*, podemos estimar un input probable a aprtir del output.

Por ejermplo, la función `bmi` .

> Si tenemos muchos datos de alturas y pesos promedios o cómo suelen ir emparejados, a partir de un bmi podríamos estimar posibles combinaciones de altura y peso que sean más probables.**Esto es lo que hace el Deep Learning y las redes neuronales**.

Funciones Irreversibles

Son aquellas que son de un sólo sentido y no hay manera práctica de obtener su inversa. Tienen una enorme importancia en la Ciencia de la Computación, especialmente en la criptografía:



- Cada vez que ves en tu browser una URL (https://keepcoding.io/blog-frr/que-es-una-url-y-una-uri/) que empieza por `https` , quiere decir que todo lo que te manda el servidor y lo que mandas tú está

Ecuación de 2º Grado

Tienes la siguiente *ecuación de segundo grado* (¿te acuerdas?):

$$y = x^2 + 3 * x - 2$$

1. Crea la función que calcula y recibe x.

¿Reversible?

Las funciones (o ecuaciones) cuadráticas (2º grado) ¿son reversibles?

🛎️ Funciones Totales y Parciales

Tenemos estas dos funciones matemáticas y tienes que crear las funciones de Python equivalentes

$$a) f(x, y) = \frac{x^2 + y^2}{x - y}$$

$$b) f(m,n) = \frac{m^2 - n^2 + n}{m * n}$$

1. Prueba ambas funciones con varios valores para asegurarte que funcionan bien.
2. ¿Hay algún valor de x o y que haga que haga que (a) estalle?
3. ¿Hay algún valor de m o n que haga que (b) estalle?
4. En la función cuadrática anterior, ¿hay algún valor de x que haga que la función estalle?

> Una función es **total** si acepta todos los posibles valores de sus parámetros.

La *función cuadrática* anterior es **total**: acepta números y traga con cualquier número posible.

> Una función es **parcial** si hay uno o más valores de sus parámetros que hacen que no pueda hacer su trabajo

Las funciones (a) y (b) son **parciales**

- Si x e y son iguales, a f(x,y) le dá un jamacuco
- Si m o n son cero, a f(m,n) le dá un jamacuco

Conclusión

Lo ideal sería que todas nuestras funciones fuesen *funciones totales*, facilitaría mucho nuestra vida.

Como bien sabrás, la vida no es fácil y habrá muchas *funciones parciales* de por medio. En esos casos,

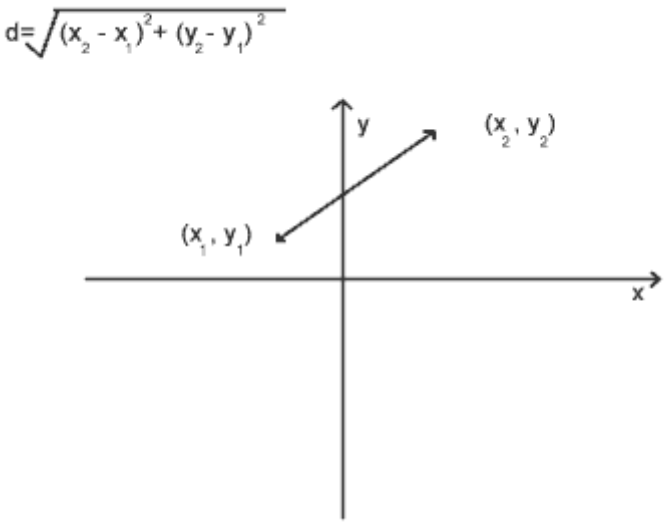
- tendremos que identificar las *condiciones excepcionales* (los valores que causan el jamacuco)
- cunado se den, tendremso que *gestionar dicho error*

La *gestión de errores* se verá con calma más adelante, pero ya eres consciete de su necesidad.

Ejercicio

Repasa todas las funciones que has creado hasta ahora y clasifícalas como *totales* o *parciales*. Si son

Distancia entre dos puntos en el plano cartesiano



Si tenemos dos puntos:

$$p_1(x_1, y_1)$$
$$p_2(x_2, y_2)$$

su *distancia cartesiana* sería la siguiente:

$$dc = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

1. Crear una función que recibe la xe y del primer punto y la x e y del segundo punto
2. Devuelve la distancia cartesiana

Curiosidad

Esta es la *distancia cartesiana*, pero hay otros tipos de distancias definidas entre puntos.

La *distancia cartesiana* es la primera que te muestra Google Maps o Waze cunado pides que te lleve a algún sitio. Normalmente cuando pides que calcule la ruta, te llevas un chasco, porque te da otra distancia mayor. ¿¿Por qué??

Porque la distancia cartesiana sólo vale si puedes volar. Si tienes que ir entre calles, tendrás que dar vueltas y eso incrementa la distancia.

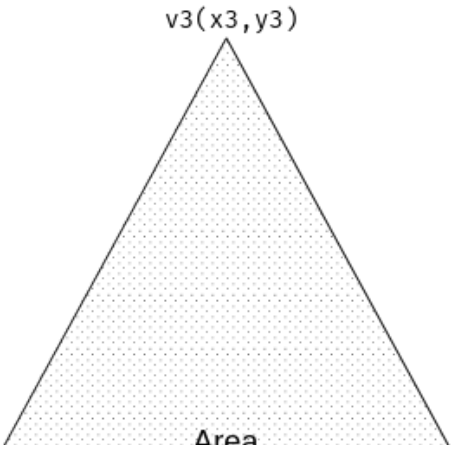
> ¿Por qué crees que Google hace eso?

Otra distancia, algo más realista para la navegación por ciudades, es la *distancia manhattan*.

Averigua en la wikipedia qué es la distancia manhattan, por qué se llama así y cómo calcularla. Crea la función Python correspondiente.

Area de un triángulo a partir de sus vértices

Si tenemos los 3 vértices de un triángulo, podemos calcular su área. Necesitaremos al función `abs` de Python



👲 🧢 Mi animal totémico está enfurecido

La función de la distancia cartesiana ya me tocó un poco los cataplines, porque necesitaba 4 parámetros para representar los dos puntos.

Ahora, resulta que necesito *seis parámetros* para los 3 cochinos vértices del triángulo de las narices.

> Es demasiado trabajo. Tiene que haber una forma mejor de hacerlo.

Abstracción al rescate

El concepto de abstracción era precisamente *empaquetar una o varias cosas, olvidarnos de sus tripas (caja negra) y tratarla como una sóla entidad*.

Esa es la salida a nuestro problema:

> Tenemos que buscar una forma de empaquetar las dos coordenadas en un solo concepto (punto) y tratarlo como una sola cosa.

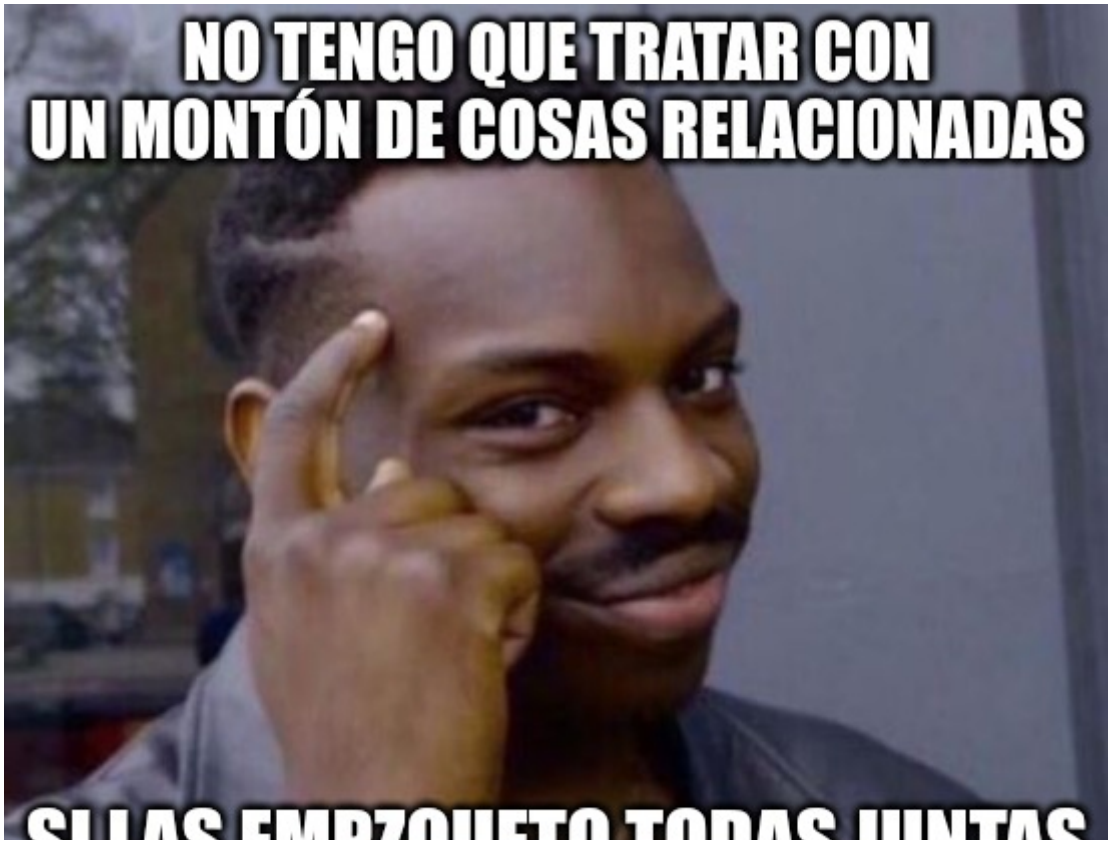
Ya hemos visto un ejemplo de algo así en Python. Recuerda los *módulos*:

1. Representan todas las definiciones globales que hay dentro de un fichero.
2. Por ejemplo el módulo `math` *aglutina* todas esas definiciones y te permite tratarlas como una sola cosa, llamada `math`
3. Para acceder a alguno de sus componentes, usamos punto: `math.pi`

No estaría de más poder hacer algo así con los puntos y acceder a ellos de la siguiente manera: `punto1.y o punto1.x`

La función `area_vert` tendría una pinta más sencilla:

```
def area_vert(v1, v2, v3):
    # Recibes los vertices que encapsulan o agrupan
    # las dos coordenadas x e y
    # Para hacer la cuentas, si necesitas acceder a una
    # de ellas, harías v1.x o v2.y, etc..
```



Abstracciones

Hemos visto dos abstraciones hasta ahora:

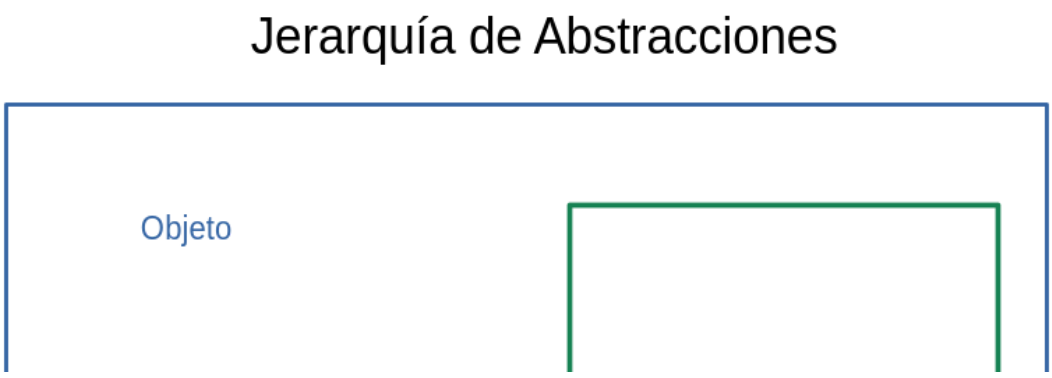
1. **Abstracción de Variable**: Pillo un dato, lo meto dentro de una cajita y le doy un nombre. Me *permite reutilizar datos*.
2. **Abstracción de Función**: Pillo una expresión, la meto en una cajita y le doy un nombre. Me *permite reutilizar expresiones*.

Ahora hemos visto una tercera,

> **Abstración de Objeto**: pillo varias variables (en ¡incluso varias funciones!), las meto en una cajita y le doy un nombre. Me permite reutilizar esos conjuntos de variables y funciones y acceda a ellas mediante un punto.

Si te fijas, una engloba a otra, como las muñecas rusas:

En el fondo, **no son más que diferentes formas de empaquetar cosas** en *cajas negras* y olvidarnos un poco de su *contenido y detalles internos*. Es decir, vagancia productiva.



Abstracción de Objeto

Más adelante ya veremos con detalle la *abstracción de objeto* con detalle. Sin embargo, ya has sentido en tus carnes la necesidad que nos ha llevado a inventarla. De momento seguiremos con la *abstracción de variable* y la *abstracción de función*.

¿Qué hemos aprendido?

- Las funciones pueden ser:
  - Reversibles
  - Irreversibles (buenas para cripto)
  - Reversibles estimables (carne de red neuronal)
  - Totales (chachi)
  - Parciales (hay que identificar los errores y gestionarlos, con calma más adelante)

Hay 3 tipos de abstraciones:

- Variable
- Función
- Objeto (veremos con calma más adelante)

Los 3 casos son formas de vaguear, hacer más con menos y olvidarnos de detalles irrelevantes

A ejercicos de funciones de cadenas