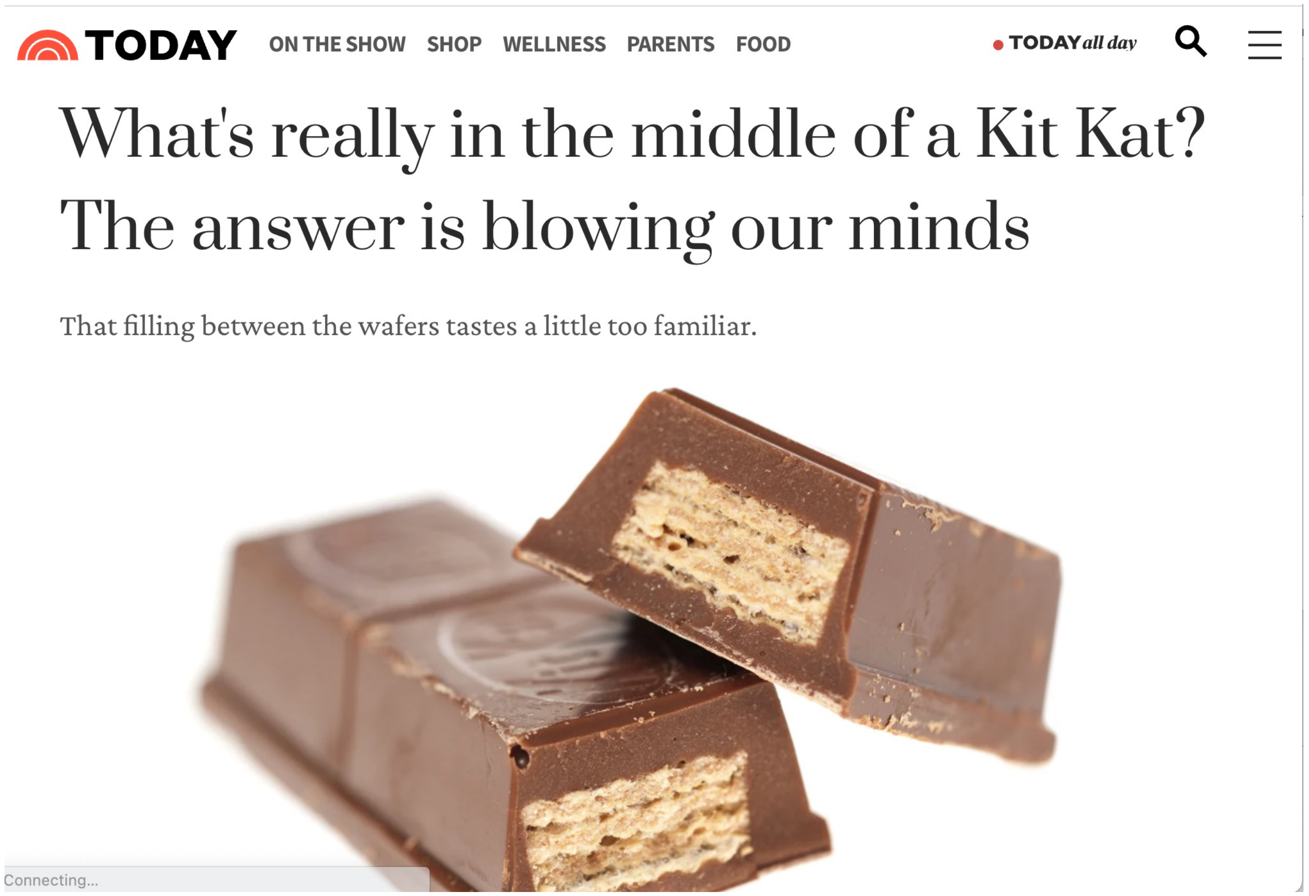
Funciones
Recursión

Clase #12

IIC1103 – Introducción a la Programación

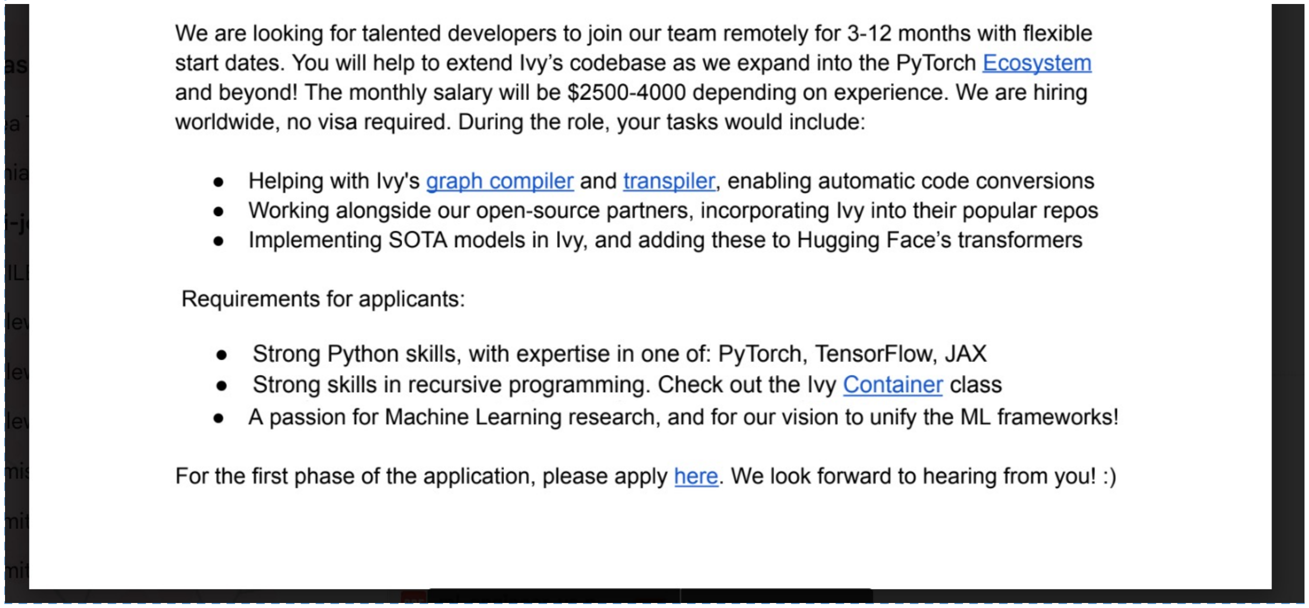


What's really in the middle of a Kit Kat? The answer is blowing our minds

That filling between the wafers tastes a little too familiar.

El plan de hoy es...

- Funciones Recursivas



We are looking for talented developers to join our team remotely for 3-12 months with flexible start dates. You will help to extend Ivy's codebase as we expand into the PyTorch Ecosystem and beyond! The monthly salary will be $2500-4000 depending on experience. We are hiring worldwide, no visa required. During the role, your tasks would include:

- Helping with Ivy's graph compiler and transpiler, enabling automatic code conversions
- Working alongside our open-source partners, incorporating Ivy into their popular repos
- Implementing SOTA models in Ivy, and adding these to Hugging Face's transformers

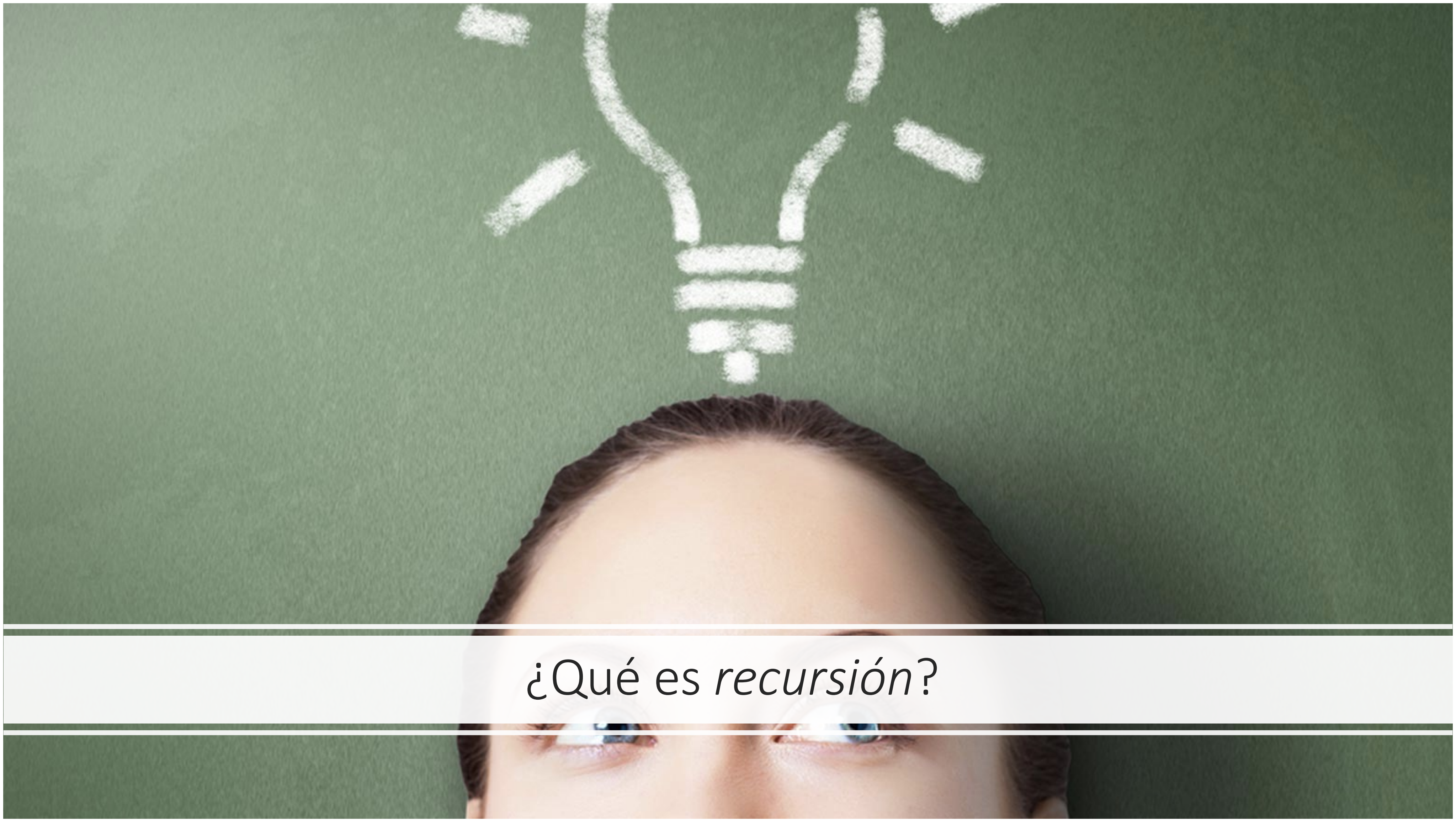
Requirements for applicants:

- Strong Python skills, with expertise in one of: PyTorch, TensorFlow, JAX
- Strong skills in recursive programming. Check out the Ivy Container class
- A passion for Machine Learning research, and for our vision to unify the ML frameworks!

For the first phase of the application, please apply here. We look forward to hearing from you! :)

Menti

- Repaso clase anterior

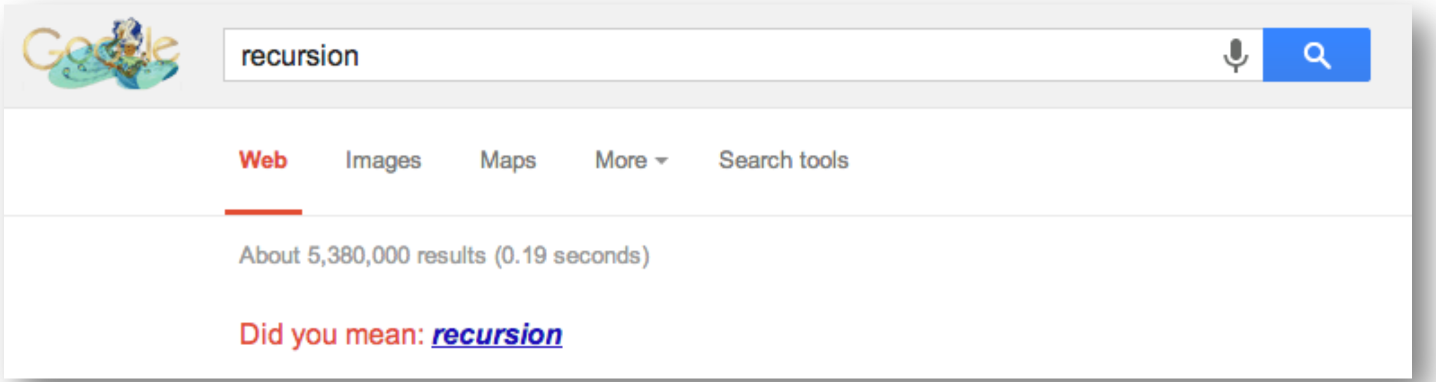


¿Qué es *recursión*?

¿Qué es *recursividad o recursión*?

- Busquemos **recursion** en Google

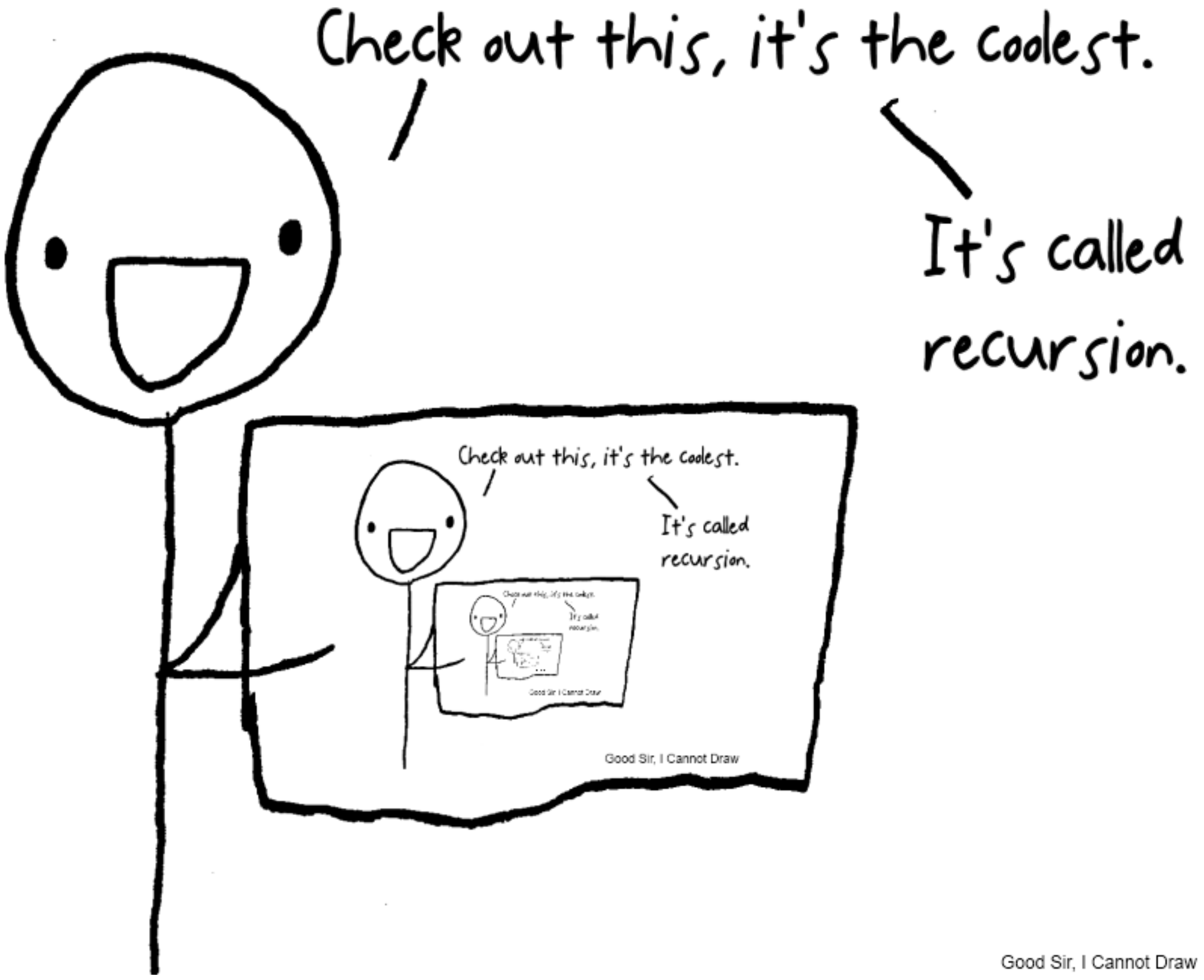
¿Qué es *recursividad* o *recursión*?


recursion

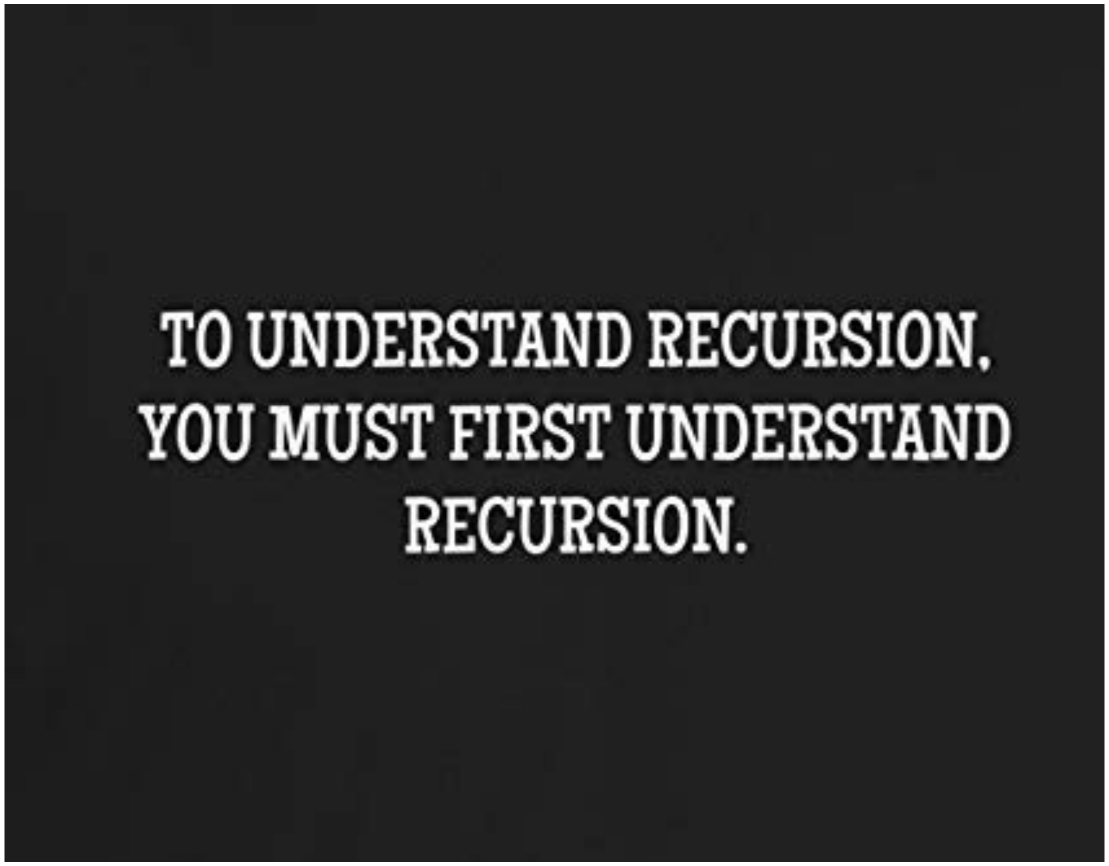
About 5,380,000 results (0.19 seconds)

Did you mean: ***recursion***

Google
recursion
Web
Images
Maps
Shopping
More
Search tools
About 1,880,000 results (0.28 seconds)
Did you mean: recursion
MC
memecenter.com/ben


Check out this, it's the coolest.
It's called recursion.
Check out this, it's the coolest.
It's called recursion.
Good Sir, I Cannot Draw
Good Sir, I Cannot Draw

¿Qué es recursión?


TO UNDERSTAND RECURSION, YOU MUST FIRST UNDERSTAND RECURSION.

PESO NETO 226g
ROYAL
BAKING POWDER
POLVO PARA HORNEAR
INDUSTRIA ARGENTINA

La respuesta:

Image Credit: anaterate via Pixabay

Kit Kats are made of Kit Kats. The manufacturers leave no Kit Kat uneaten, so every reject is mashed into a paste that's used to fill the wafers. Because every imperfect Kit Kat was already filled with imperfect Kit Kats, and some of the Kit Kats they'll fill will also be rejected, every time you eat a Kit Kat you're basically eating layers of Kit Kats within Kit Kats.

SAFELY ENDANGERED
SWEET JESUS, POOH! THAT'S NOT HONEY
YOU'RE EATING RECURSION
SAFELY ENDANGERED
SWEET JESUS, POOH! THAT'S NOT HONEY
YOU'RE EATING RECURSION
ifunny.co

Recursión

- “Resolver un problema mediante recursión significa que la solución depende de las soluciones de pequeñas instancias del mismo problema” (Wikipedia)

Problema #1: Factorial

- Usando la siguiente definición:
  - n! = 1*2*3*…*(n-1)*n
  - 0!=1

```
• def factorial(n):
•     producto=1
•     for i in range(1,n+1):
•         producto*=i
•     return producto
```

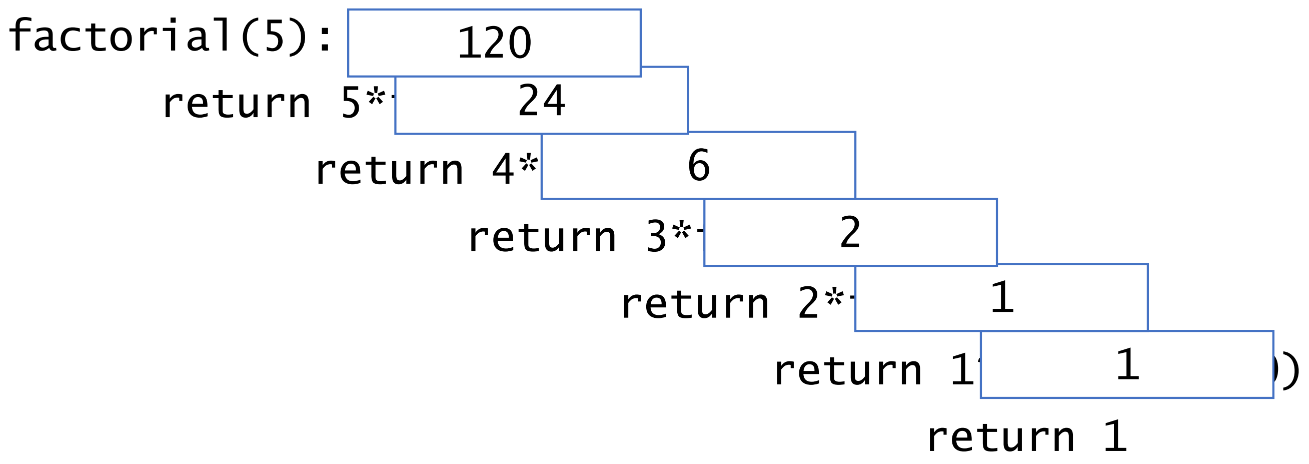
Problema #1b: Factorial

- Usando la siguiente definición:
  - n! = n*(n-1)!
  - 0!=1

Solución: usando recursión

```
• def factorial(n):
•     if n==0:
•         return 1
•     return n*factorial(n-1)
```

¿Cómo se calcula factorial(5)?

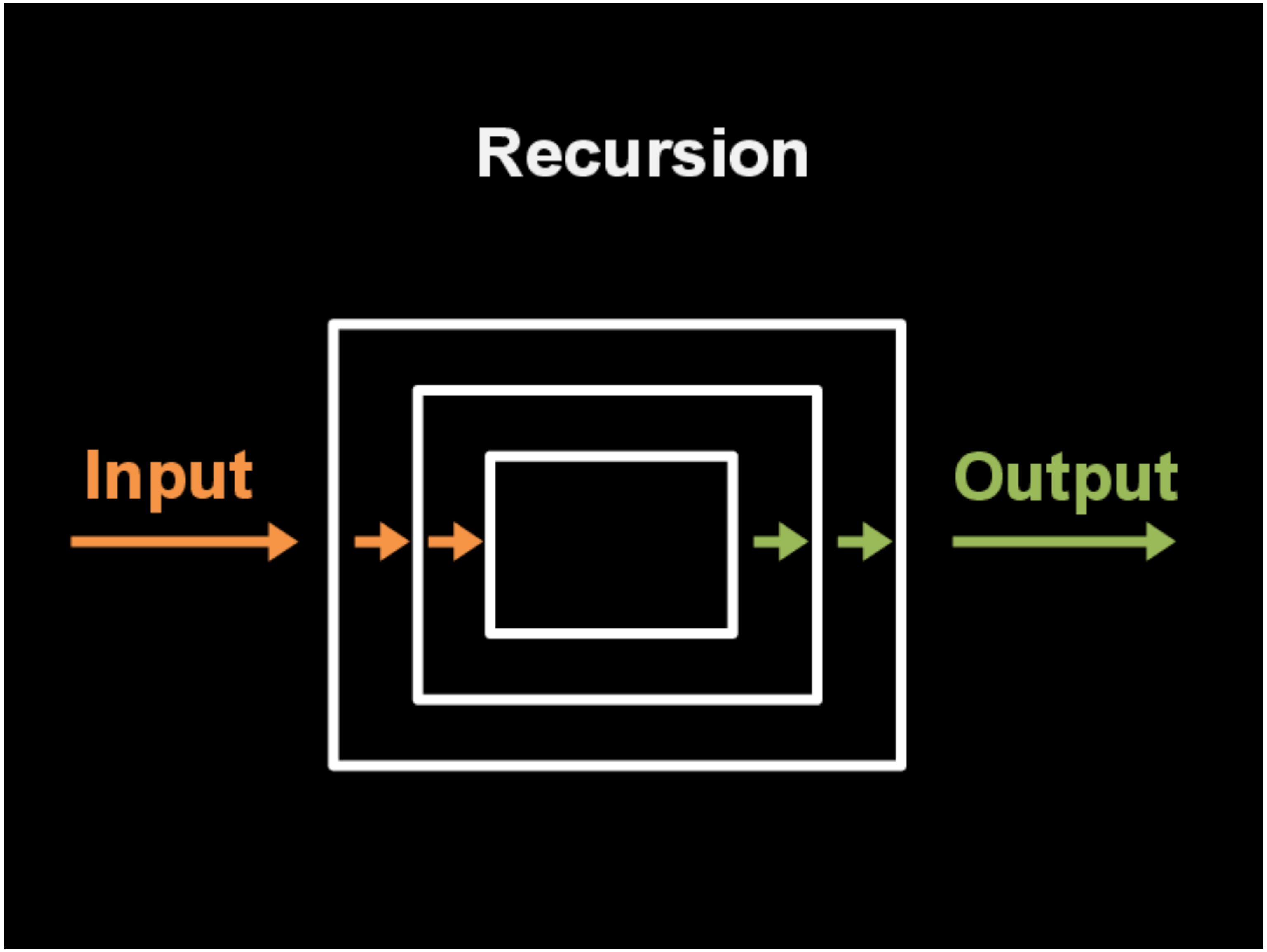


```
factorial(5):    120
    return 5*         24
        return 4*            6
            return 3*             2
                return 2*              1
                    return 1               1        )
                        return 1
```

Función recursiva

- Debe tener una salida no recursiva (caso base)

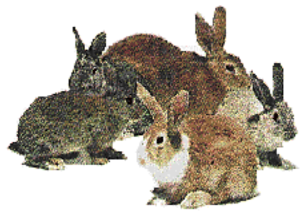
- Se invoca a sí mismo
  - Las llamadas recursivas deben converger al caso base, es decir, se debe ir reduciendo el tamaño del problema


Recursion
Input
Output

Recursion – achicando el tamaño del problema?

Problema 2: Los conejos de Fibonacci

The original problem that Fibonacci investigated (in the year 1202) was about how fast rabbits could breed in ideal circumstances.



Suppose a newly-born pair of rabbits, one male, one female, are put in a field. Rabbits are able to mate at the age of one month so that at the end of its second month a female can produce another pair of rabbits. Suppose that our rabbits **never die** and that the female **always** produces one new pair (one male, one female) **every month** from the second month on. The puzzle that Fibonacci posed was...

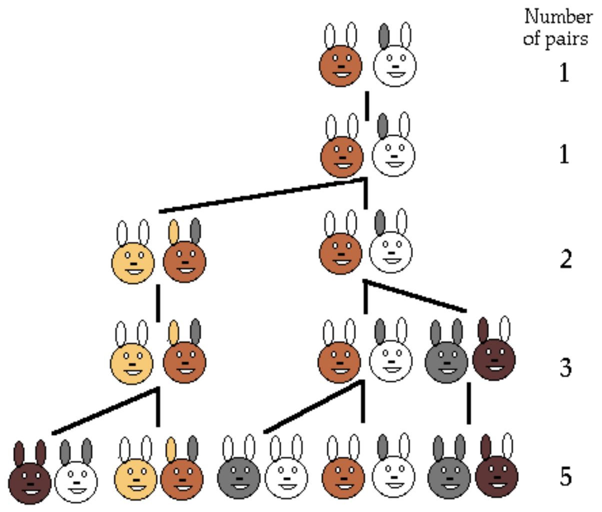
How many pairs will there be in one year?

1. At the end of the first month, they mate, but there is still one only 1 pair.
2. At the end of the second month the female produces a new pair, so now there are 2 pairs of rabbits in the field.
3. At the end of the third month, the original female produces a second pair, making 3 pairs in all in the field.
4. At the end of the fourth month, the original female has produced yet another new pair, the female born two months ago produces her first pair also, making 5 pairs.

http://www.maths.surrey.ac.uk/hosted-sites/R.Knott/Fibonacci/fibnat.html#Rabbits

Problema 2: Los conejos de Fibonacci

1. At the end of the first month, they mate, but there is still one only 1 pair.
2. At the end of the second month the female produces a new pair, so now there are 2 pairs of rabbits in the field.
3. At the end of the third month, the original female produces a second pair, making 3 pairs in all in the field.
4. At the end of the fourth month, the original female has produced yet another new pair, the female born two months ago produces her first pair also, making 5 pairs.



Problema #2: Fibonacci

- Escribir una función/método que calcule el iésimo número de Fibonacci

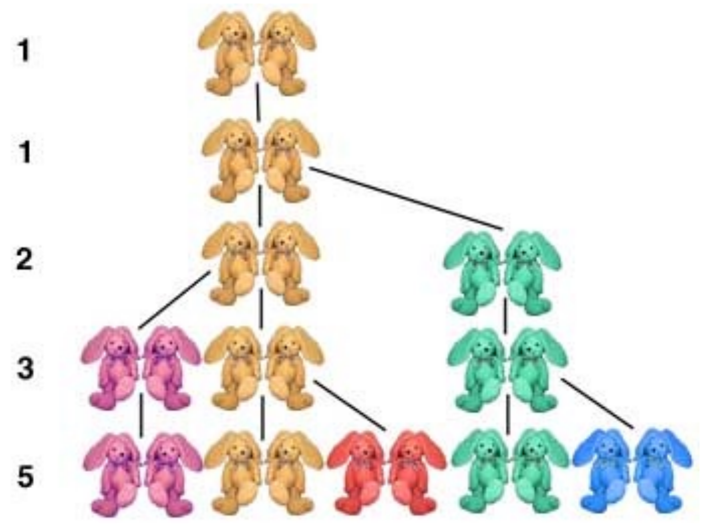
- Fibonacci:
- `1, 1, 2, 3, 5, 8, 13, 21, 34, 55, …`

1er mes→

2do mes→

3er mes→

…. →



Solución

- Caso Base:
  - i = 0 → 0
  - i = 1 → 1
- Caso Recursivo:
  - fibonacci(i) = fibonacci(i-1)+fibonacci(i-2)

Solución

```
def fibonacci(i):
    if i==0 or i==1:
        return i
    return fibonacci(i-1)+fibonacci(i-2)
```

¿Qué pasa?

- `def fibonacci(i):`
- `    return fibonacci(i-1)+fibonacci(i-2)`
-

No hay caso base, ¡recursión nunca termina!

Ventajas/Desventajas de métodos recursivos

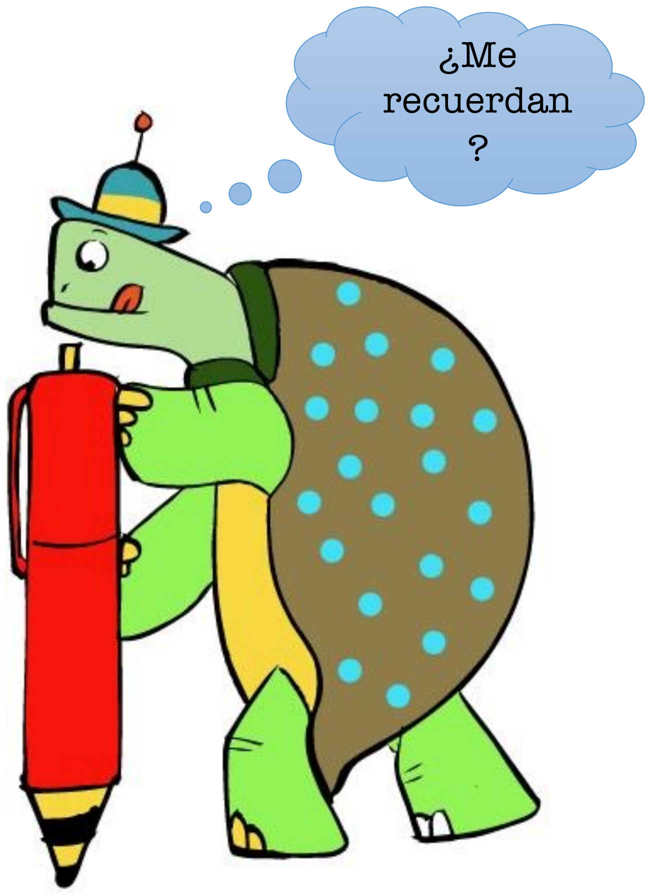
- Se usa más memoria
- Lentitud (más llamadas)
- Para ciertos problemas, es más fácil y elegante de programar que la solución iterativa
  - Por lo general, existe una alternativa usando ciclos
    - (Puede ser bastante difícil de programar)

Problema #3: turtle

- `from turtle import *`

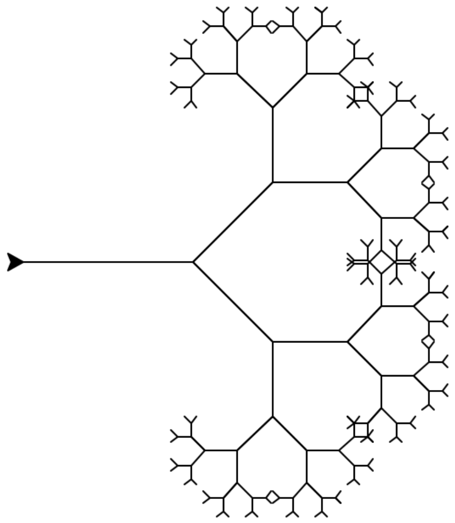
- `forward(num),`
- `back(num),`
- `left(num),`
- `right(num), etc`

- **`dibujar_arbol(l,n)`**

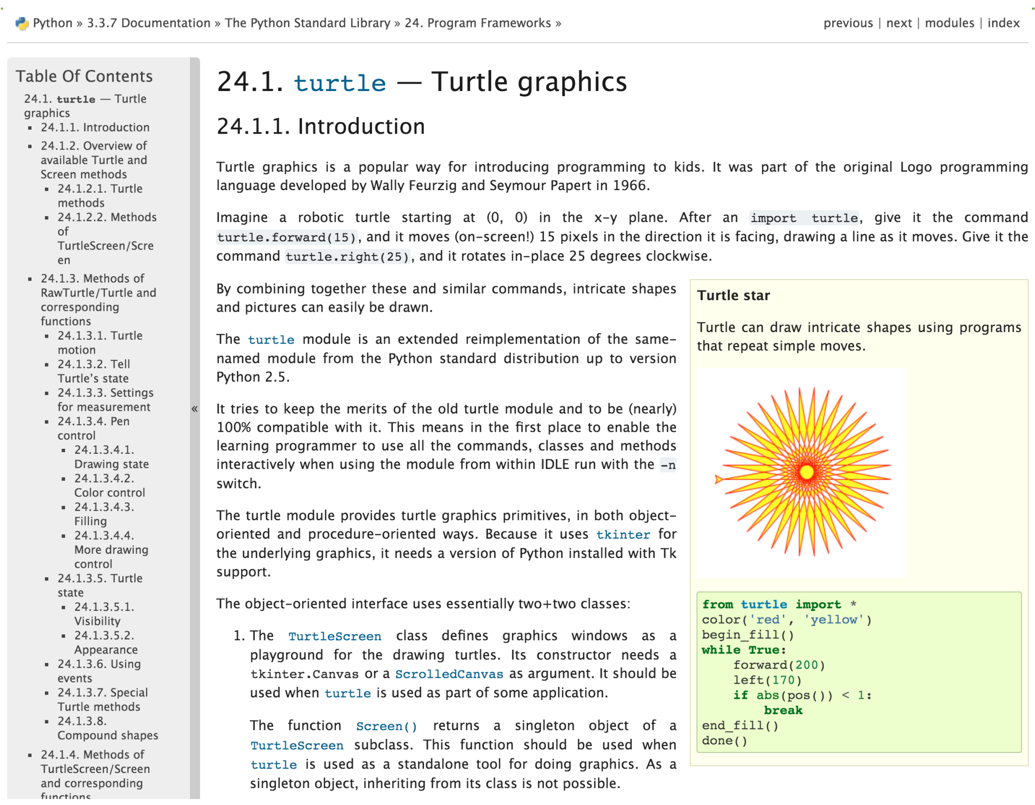


https://docs.python.org/3.3/library/turtle.html

Dibujamos esto:



¿Investigar más?



Python » 3.3.7 Documentation » The Python Standard Library » 24. Program Frameworks »

previous | next | modules | index

24.1. `turtle` — Turtle graphics

24.1.1. Introduction

Turtle graphics is a popular way for introducing programming to kids. It was part of the original Logo programming language developed by Wally Feurzig and Seymour Papert in 1966.

Imagine a robotic turtle starting at (0, 0) in the x-y plane. After an `import turtle`, give it the command `turtle.forward(15)`, and it moves (on-screen!) 15 pixels in the direction it is facing, drawing a line as it moves. Give it the command `turtle.right(25)`, and it rotates in-place 25 degrees clockwise.

By combining together these and similar commands, intricate shapes and pictures can easily be drawn.

Turtle star

Turtle can draw intricate shapes using programs that repeat simple moves.

```python
from turtle import *
color('red', 'yellow')
begin_fill()
while True:
    forward(200)
    left(170)
    if abs(pos()) < 1:
        break
end_fill()
done()
```

The `turtle` module is an extended reimplementation of the same-named module from the Python standard distribution up to version Python 2.5.

It tries to keep the merits of the old turtle module and to be (nearly) 100% compatible with it. This means in the first place to enable the learning programmer to use all the commands, classes and methods interactively when using the module from within IDLE run with the `-n` switch.

The turtle module provides turtle graphics primitives, in both object-oriented and procedure-oriented ways. Because it uses `tkinter` for the underlying graphics, it needs a version of Python installed with Tk support.

The object-oriented interface uses essentially two+two classes:

1. The `TurtleScreen` class defines graphics windows as a playground for the drawing turtles. Its constructor needs a `tkinter.Canvas` or a `ScrolledCanvas` as argument. It should be used when `turtle` is used as part of some application.

   The function `Screen()` returns a singleton object of a `TurtleScreen` subclass. This function should be used when `turtle` is used as a standalone tool for doing graphics. As a singleton object, inheriting from its class is not possible.

https://docs.python.org/3.3/library/turtle.html

Solución

-
  ```
  from turtle import *

  def dibujar_arbol(largo,n):
      if n==0:
              return
      forward(largo)
      left(30)
      dibujar_arbol(largo//1.5,n-1)
      right(60)
      dibujar_arbol(largo//1.5,n-1)
      left(30)
      back(largo)

  dibujar_arbol(100,5)
  input()
  ```

Práctica

- Escribir función ***RECURSIVA*** que calcule x^y (con y >= 0):

- `def potencia(x,y):`

- `Recordar que debe haber un caso base y un caso recursivo`

Práctica

- Escribe una función ***RECURSIVA*** `unoytres(n)` que imprima todos los números menores o iguales a n que solo consisten de los dígitos 1 y 3.
- Ejemplo:
- `unoytres(20)` imprime: `13 11 3 1`
- `unoytres(40)` imprime: `33 31 13 11 3 1`
- `unoytres(3)` imprime: `3 1`

Resumen de hoy

- **Hoy vimos**:
- Funciones recursivas: Caso base + caso recursivo (la función se llama a sí misma)