Python a Través de un ejemplo

Irene Pérez Encinar

@irenuchi

Mi historia de amor con Python

```
>> Aprendiz de todo, experta en nada
```

Mi historia de amor con Python

>> Aprendiz de todo, experta en nada

>> Primer contacto - robótica
   >> ¿dónde están mis {};?
   >> ¿tabuladores o espacios?

Mi historia de amor con Python

>> Aprendiz de todo, experta en nada

>> Primer contacto - robótica
   >> ¿dónde están mis {};?
   >> ¿tabuladores o espacios?

>> Aprender desde cero para enseñarlo <3 <3
   >> cada lenguaje tiene su forma de hacer las cosas
   >> Buen lenguaje para aprender
   >> CodeCombat
   >> PythonLearn

Lo que me encanta de Python...

>> Sintaxis que favorece código legible
   >> indentaciones - not so bad

Lo que me encanta de Python...

>> Sintaxis que favorece código legible
   >> indentaciones - not so bad

>> Poca curva de aprendizaje (“batteries included”)

Lo que me encanta de Python...

>> Sintaxis que favorece código legible
   >> indentaciones - not so bad

>> Poca curva de aprendizaje (“batteries included”)

>> Compacto y rápido de programar: pocas líneas de código
   >> Me permite trabajar de forma más eficiente

Lo que me encanta de Python...

>> Sintaxis que favorece código legible
   >> indentaciones - not so bad

>> Poca curva de aprendizaje (“batteries included”)

>> Compacto y rápido de programar: pocas líneas de código
   >> Me permite trabajar de forma más eficiente

>> Consola: o cómo nunca más usar calculadora ;)

Lo que me encanta de Python...

>> Sintaxis que favorece código legible
   >> indentaciones - not so bad

>> Poca curva de aprendizaje (“batteries included”)

>> Compacto y rápido de programar: pocas líneas de código
   >> Me permite trabajar de forma más eficiente

>> Consola: o cómo nunca más usar calculadora ;)

>> Comunidad de usuarios

Lo que me encanta de Python...

>> Sintaxis que favorece código legible
   >> indentaciones - not so bad

>> Poca curva de aprendizaje (“batteries included”)

>> Compacto y rápido de programar: pocas líneas de código
   >> Me permite trabajar de forma más eficiente

>> Consola: o cómo nunca más usar calculadora ;)

>> Comunidad de usuarios

>> Se llama así por los Monty Python :)

Vale, pero... ¿para qué?

>> Cifras y Letras

   >> Programa que juegue a las “letras”

Vale, pero... ¿para qué?

>> Cifras y Letras
   >> Programa que juegue a las “letras”

>> Lemario del español
   >> Cada palabra en una línea

Vale, pero... ¿para qué?

>> Cifras y Letras
   >> Programa que juegue a las “letras”

>> Lemario del español
   >> Cada palabra en una línea

```
f = open("lemario-general-del-espanol.txt")
words = f.read()
```

¿Podemos hacerlo mejor?

```
f = open("lemario-general-del-espanol.txt")
try:
    words = f.read()
except:
    print("Ooops!")
finally:
    f.close()
```

¿Podemos hacerlo mejor?

```
with open("lemario-general-del-espanol.txt") as f:
    words = f.read()
```

¿Podemos hacerlo mejor?

```
with open("lemario-general-del-espanol.txt") as f:
    words = f.read()
```

>> *words* es un poco inmanejable, ¿no?

¿Podemos hacerlo mejor?

```
with open("lemario-general-del-espanol.txt") as f:
    words = f.read()
```

>> *words* es un poco inmanejable, ¿no?

```
words = f.readlines()
```

```
words = list(f)
```

¿Podemos hacerlo mejor?

```
>> Tendremos que quitar el \n
```

¿Podemos hacerlo mejor?

>> Tendremos que quitar el \n

```
with open("lemario-general-del-espanol.txt") as f:
    words = f.readlines()
    clean_words = []
    for word in words:
        clean_words.append(word.strip())
```

¿Podemos hacerlo mejor?

>> Tendremos que quitar el \n

```
with open("lemario-general-del-espanol.txt") as f:
    words = [word.strip() for word in f.readlines()]
```

Volviendo al problema que nos ocupa...

```
letters = ['o','p','d','e','t','o','r']
```

Volviendo al problema que nos ocupa...

```
letters = ['o','p','d','e','t','o','r']
```

```
from itertools import permutations

letters = ['o','p','d','e','t','o','r']
permutations(letters, 7)
```

Volviendo al problema que nos ocupa...

```
letters = ['o','p','d','e','t','o','r']
```

```
from itertools import permutations

letters = ['o','p','d','e','t','o','r']
permutations(letters, 7)
```

>> ¿Nos interesan todas las posibles permutaciones de las letras?

Volviendo al problema que nos ocupa...

>> De todas las permutaciones, sólo nos interesan las que están en el lemario

```
words_7l = []
for p in permutations(letters, 7):
    if p in words:
        words_7l.append(p)
```

Volviendo al problema que nos ocupa...

>> De todas las permutaciones, sólo nos interesan las que están en el lemario

```
words_7l = [p for p in permutations(letters, 7) if p in words]
```

Volviendo al problema que nos ocupa...

>> De todas las permutaciones, sólo nos interesan las que están en el lemario

```
words_7l = [p for p in permutations(letters, 7) if p in words]
```

>> ¿Esto funciona?

Volviendo al problema que nos ocupa...

```
words_7l = ["".join(p) for p in permutations(letters, 7) if "".join(p) in words]
```

Volviendo al problema que nos ocupa...

```
words_7l = ["".join(p) for p in permutations(letters, 7) if "".join(p) in words]
```

>> Si una letra se repite… hay dos permutaciones de la palabra

Volviendo al problema que nos ocupa...

```
words_7l = ["".join(p) for p in permutations(letters, 7) if "".join(p) in words]
```

>> Si una letra se repite… hay dos permutaciones de la palabra

```
words_7l = set(["".join(p) for p in permutations(letters, 7) if "".join(p) in words])
```

Volviendo al problema que nos ocupa...

>> Nos quedaría obtener las palabras de 6, 5, 4 letras...

Volviendo al problema que nos ocupa...

>> Nos quedaría obtener las palabras de 6, 5, 4 letras...

```
possible_answers = dict()
for i in range(7, 2, -1):
    possible_answers[i] = set(["".join(p) for p in permutations(letters, i)
                               if "".join(p) in words])
```

Volviendo al problema que nos ocupa...

>> Nos quedaría obtener las palabras de 6, 5, 4 letras...

```
possible_answers = {i : set(["".join(p) for p in permutations(letters, i) if "".join(p) in words]) for i in range(7, 2, -1)}
```

```
possible_answers = {i : set(["".join(p) for p in permutations(letters, i) if "".join(p) in words])
                    for i in range(7, 2, -1)}
```

Y así quedaría nuestro programa completo...

```
from itertools import permutations

letters = ['o','p','d','e','t','o','r']

with open("lemario-general-del-espanol.txt") as f:
    words = [word.strip() for word in f.readlines()]

possible_answers = {i : set(["".join(p) for p in permutations(letters, i) if "".join(p) in words])
                    for i in range(7, 2, -1)}
```

Una "pequeña" optimización

```
from itertools import permutations

letters = ['o','p','d','e','t','o','r']

with open("lemario-general-del-espanol.txt") as f:
    words = set([word.strip() for word in f.readlines()])

possible_answers = {i : set(["".join(p) for p in permutations(letters, i) if "".join(p) in words])
                   for i in range(7, 2, -1)}
```

BONUS: SOLUCIÓN ALTERNATIVA

Permutaciones de	Combinaciones de	7 elementos tomados de...
5040	1	7 en 7
5040	7	6 en 6
2520	21	5 en 5
840	35	4 en 4
210	35	3 en 3

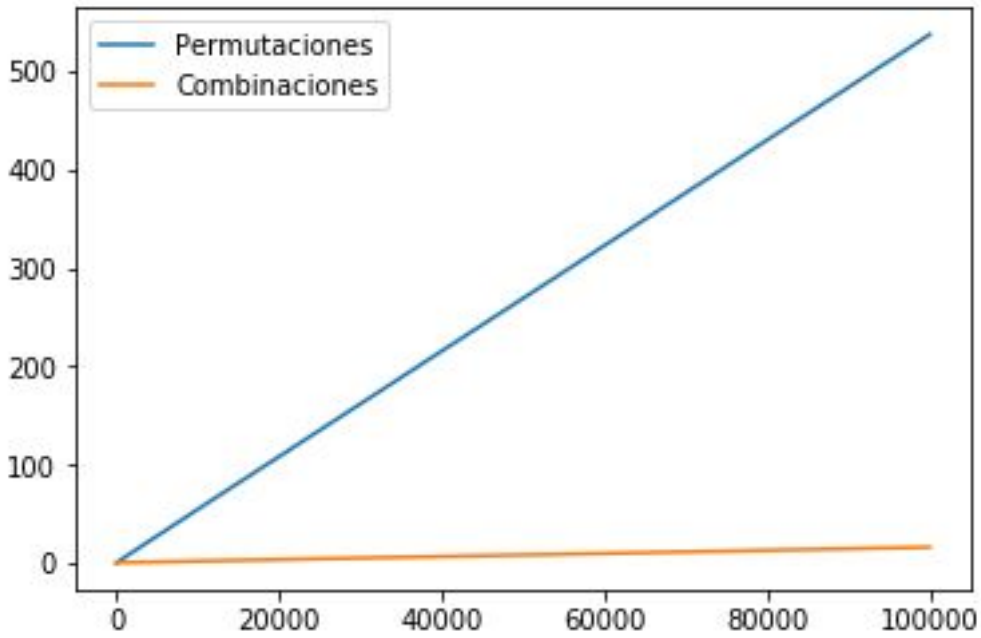
BONUS: SOLUCIÓN ALTERNATIVA

```
from itertools import combinations
from collections import defaultdict

letters = ['o','p','d','e','t','o','r']

words = defaultdict(list)
with open("lemario-general-del-espanol.txt") as f:
    for word in f.readlines():
        words["".join(sorted(word.strip()))].append(word.strip())

possible_answers = {i : [words["".join(sorted(c))] for c in combinations(letters, i) if "".join(sorted(c)) in words]
                    for i in range(7, 2, -1)}
```

BONUS: COMPARATIVA

Iteraciones	Tiempo permutaciones	Tiempo combinaciones
10	0.13	0.35
100	0.67	0.38
1000	5.75	0.5
10000	54.56	1.89
100000	537.6	16.07

BONUS: COMPARATIVA



¡Gracias!

Irene Pérez Encinar @irenuchi

https://github.com/IrenePEncinar/workshops