```
def repaso_examen():
    print('Introducción a la Programación')
```

Strings

```
nuevo_string = 'Se pasa Intro a la Progra?'
otro_string = "Obvio que si!!"
```

```
string_largo = ('Esto es un string bastante largo. Por alguna razón, lo estoy '
                'escribiendo a las 3 A.M.\nAcabo de saltarme una línea, pero no '
                'sé por qué lo hice.\nSólo lo hice.\nMe pregunto cuántos de '
                'ustedes estarán despiertos a esta hora.')
```

Output: Esto es un string bastante largo. Por alguna razón, lo estoy escribiendo a las 3 A.M.
Acabo de saltarme una línea, pero no sé por qué lo hice.
Sólo lo hice.
Me pregunto cuántos de ustedes estarán despiertos a esta hora.

```
nuevo_string = 'Se pasa la analogía de Intro a la Progra?'
nuevo_string[0]  # 'S'
nuevo_string[3]  # 'p'
nuevo_string[-1]  # '?'
nuevo_string[-3]  # 'r'
nuevo_string[11:14]  # 'ana'
```

```
nuevo_string = 'HOLA GENTE'
for letra in nuevo_string:
  print(letra)
```

Output:

```
'H'
'O'
'L'
'A'
' '
'G'
'E'
'N'
'T'
'E'
```

```python
nuevo_string = 'FLIPITY-FLAPITY-FLUPITY'
lista_rara = nuevo_string.split('-')
print(lista_rara)  # ['FLIPITY', 'FLAPITY', 'FLUPITY']
```

Listas

```python
una_lista_cualquiera = ['una', 'lista', 'cualquiera']  # creatividad nivel 9999
```

```
otra_lista_cualquiera = [['una', 'lista', 'cualquiera'],
                         ['dos', 'listas', 'cualquiera'],
                         ['suerte', 'en el', 'examen']]
```

```
.append(elemento)
.remove(elemento)
.pop(indice)
.insert(indice, elemento)
```

```python
una_lista_cualquiera = ['una', 'lista', 'cualquiera']
una_lista_cualquiera[0]  # 'una'
una_lista_cualquiera[-1]  # 'cualquiera'
una_lista_cualquiera[1]  # 'lista'
una_lista_cualquiera[:-1]  # ['una', 'lista']
una_lista_cualquiera[0:2]  # ['una', 'lista']
```

Programación Orientada a Objetos

```
class Persona:
    pass
```

```python
class Persona:

    def __init__(self, nombre_externo):
        self.nombre = nombre_externo
        self.especie = 'Humano'
```

```python
class Persona:

    def __init__(self, nombre_externo):
        self.nombre = nombre_externo
        self.especie = 'Humano'


persona_1 = Persona('Daniel')
```

```python
class Persona:

    def __init__(self, nombre_externo):
        self.nombre = nombre_externo
        self.especie = 'Humano'


persona_1 = Persona('Daniel')
persona_2 = Persona('Bryan')
```

Los objetos también pueden tener atributos que sean otros objetos!

```python
class GrupoAmigos:

    def __init__(self, nombre_grupo, lista_amigos):
        self.nombre = nombre_grupo
        self.amigos = lista_amigos
```

```python
class GrupoAmigos:
    def __init__(self, nombre_grupo, lista_amigos):
        self.nombre = nombre_grupo
        self.amigos = lista_amigos

    def mostrar_nombres_amigos(self):
        for amigo in self.amigos:
            print(amigo.nombre)

class Persona:
    def __init__(self, nombre_externo):
        self.nombre = nombre_externo
        self.especie = 'Humano'

persona_1 = Persona('Daniel')
persona_2 = Persona('Bryan')
persona_3 = Persona('Gonzalo')
persona_4 = Persona('Pelao')
amigos = [persona_1, persona_2, persona_3, persona_4]

grupo_amigos = GrupoAmigos('Las Rocas', amigos)
```

```python
class GrupoAmigos:
    def __init__(self, nombre_grupo, lista_amigos):
        self.nombre = nombre_grupo
        self.amigos = lista_amigos

    def mostrar_nombres_amigos(self):
        for amigo in self.amigos:
            print(amigo.nombre)

class Persona:
    def __init__(self, nombre_externo):
        self.nombre = nombre_externo
        self.especie = 'Humano'

persona_1 = Persona('Daniel')
persona_2 = Persona('Bryan')
persona_3 = Persona('Gonzalo')
persona_4 = Persona('Pelao')
amigos = [persona_1, persona_2, persona_3, persona_4]

grupo_amigos = GrupoAmigos('Las Rocas', amigos)
grupo_amigos.mostrar_nombres_amigos()
```

```python
class Persona:

    def __init__(self, nombre_externo):
        self.nombre = nombre_externo
        self.especie = 'Humano'
        self.apodo = None

    def cambiar_apodo(self, nuevo_apodo):
        self.apodo = nuevo_apodo

persona_1 = Persona('Bryan')
persona_2 = Persona('John Cena')
persona_1.cambiar_apodo('Guatón Byron')
persona_2.cambiar_apodo('Tinky Winky')
```

Lo anterior también se podría hacer sin tener un método específico para ello!!

```
class Persona:

    def __init__(self, nombre_externo):
        self.nombre = nombre_externo
        self.especie = 'Humano'
        self.apodo = None


persona_1 = Persona('Bryan')
persona_2 = Persona('John Cena')
persona_1.apodo = 'Guatón Byron'
persona_2.apodo = 'Tinky Winky'
```

Recursión y BackTracking

```
def recursion():
    recursion()
```

Esta función nunca terminará!!

```python
def recursion(numero):
    if numero <= 0:
        return
    print(numero)
    recursion(numero - 1)

recursion(3)
```

Qué pasará?

```
3
2
1
```

Idea General

Queremos llamar la función con algunos argumentos y chequear si tenemos lo necesario para detener la función (caso base). Si ya tenemos lo que queremos, retornamos algo. En otro caso, ejecutamos la función de nuevo (generalmente se alteran levemente los argumentos) y se vuelve a llamar a la función con los argumentos levemente alterados.