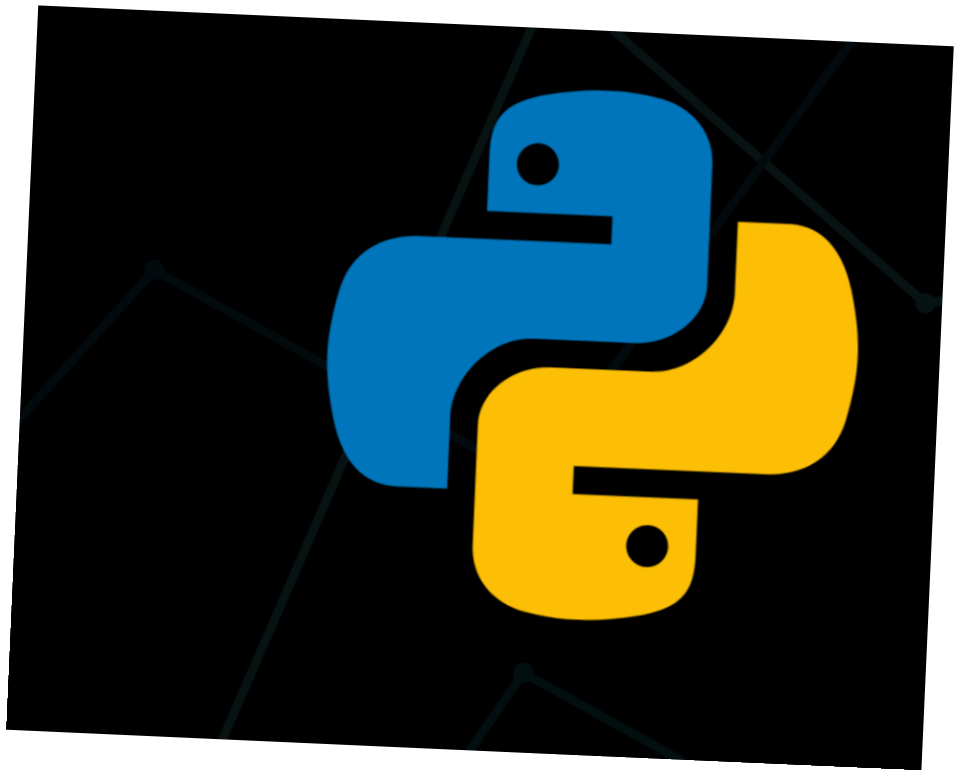
for e range() — Normalização de dados, paginação e fluxos de processo



Programando em Python.

O que é o for em Python

O ***for*** percorre, elemento a elemento, qualquer **iterável** (lista, tupla, string, dicionário — por padrão, chaves —, arquivo linha a linha, etc.).
Modelo mental: “para cada elemento em sequencia, faça…”.

Sintaxe:

```
for elemento in sequencia:
# usa 'elemento' aqui
```

A variável elemento recebe **um valor por vez.**

Não precisa incrementar manualmente; o for **avança sozinho** quando o iterável termina.

O que é range()

range() cria uma sequência de inteiros sob demanda (eficiente em memória). É muito usado junto do for quando você precisa repetir N vezes ou trabalhar por índice/janelas.

Formas:

- range(fim) → 0, 1, 2, ..., fim-1
- range(inicio, fim) → inicio, ..., fim-1
- range(inicio, fim, passo) → pula de passo em passo (positivo ou negativo) até antes de fim

Atenção: o **fim é exclusivo**. Essa é a fonte nº 1 de erro com range().

Exemplos rápidos:

```
range(5)          # 0,1,2,3,4
range(1, 5)       # 1,2,3,4
range(10, 0, -2) # 10,8,6,4,2
```

Cadência de códigos — apenas for

Normalização de dados, checagem de e-mail, reconciliação simples de estoque e saneamento de cadastro. Aqui o for percorre coleções diretamente.

python.py > ...

```python
# Objetivo: padronizar nomes para busca e exibição
produtos = ["  mouse óptico  ", "Teclado MECÂNICO", "  CABO HDMI 2m", "Notebook  "]

for nome in produtos:
    nome_limpo = nome.strip()
    nome_busca = nome_limpo.lower()
    nome_exibicao = nome_limpo.title()
    print(f"Original: {nome} | Busca: {nome_busca} | Exibição: {nome_exibicao}")
```

PROBLEMAS SAÍDA CONSOLE DE DEPURAÇÃO TERMINAL PORTAS

Python: python

```
PS C:\Users\#_Ohana_#\Documents\Projeto_Git> & C:/Users/#_Ohana_#/AppData/Local/Programs/Python/Python313/python.exe c:/Users/#_Ohana_#/Documents/Projeto_Git/python.py
Original:   mouse óptico   | Busca: mouse óptico | Exibição: Mouse Óptico
Original: Teclado MECÂNICO | Busca: teclado mecânico | Exibição: Teclado Mecânico
Original:   CABO HDMI 2m | Busca: cabo hdmi 2m | Exibição: Cabo Hdmi 2M
Original: Notebook   | Busca: notebook | Exibição: Notebook
PS C:\Users\#_Ohana_#\Documents\Projeto_Git>
```

```python
# Objetivo: padronizar nomes para busca e exibição
produtos = ["  mouse óptico  ", "Teclado MECÂNICO", "   CABO HDMI 2m", "Notebook   "]  # Cria uma lista com nomes de produtos, alguns com espaços e letras maiúsculas misturadas

for nome in produtos:  # Percorre cada item da lista, armazenando o valor atual na variável 'nome'
    nome_limpo = nome.strip()  # Remove espaços extras no início e no final do texto
    nome_busca = nome_limpo.lower()  # Converte todas as letras para minúsculas, ideal para buscas padronizadas
    nome_exibicao = nome_limpo.title()  # Converte o texto para o formato de título (primeira letra maiúscula de cada palavra)
    print(f"Original: {nome} | Busca: {nome_busca} | Exibição: {nome_exibicao}")  # Exibe todas as versões: original, busca e exibição

```

```python
# Objetivo: contar provedores de e-mail mais comuns
emails = [
    "ana@gmail.com", "joao@outlook.com", "suporte@empresa.com.br",
    "vendas@yahoo.com", "contato@gmail.com", "RH@OUTLOOK.COM"
]

gmail = outlook = yahoo = outros = 0

for e in emails:
    e_norm = e.strip().lower()
    if "@gmail.com" in e_norm:
        gmail += 1
    elif "@outlook.com" in e_norm or "@hotmail.com" in e_norm:
        outlook += 1
    elif "@yahoo.com" in e_norm:
        yahoo += 1
    else:
        outros += 1

print(f"Gmail: {gmail} | Outlook/Hotmail: {outlook} | Yahoo: {yahoo} | Outros: {outros}")
```

```python
# Objetivo: contar provedores de e-mail mais comuns
emails = [  # Cria uma lista com endereços de e-mail variados
    "ana@gmail.com", "joao@outlook.com", "suporte@empresa.com.br",
    "vendas@yahoo.com", "contato@gmail.com", "RH@OUTLOOK.COM"
]

gmail = outlook = yahoo = outros = 0  # Cria quatro contadores, iniciando todos com o valor 0

for e in emails:  # Percorre cada e-mail da lista
    e_norm = e.strip().lower()  # Remove espaços no início/fim e transforma tudo em minúsculas
    if "@gmail.com" in e_norm:  # Verifica se o e-mail contém o domínio do Gmail
        gmail += 1  # Se sim, soma 1 ao contador de Gmail
    elif "@outlook.com" in e_norm or "@hotmail.com" in e_norm:  # Verifica se é Outlook ou Hotmail
        outlook += 1  # Soma 1 ao contador de Outlook/Hotmail
    elif "@yahoo.com" in e_norm:  # Verifica se é Yahoo
        yahoo += 1  # Soma 1 ao contador de Yahoo
    else:  # Caso não seja nenhum dos anteriores
        outros += 1  # Soma 1 ao contador de outros provedores
print(f"Gmail: {gmail} | Outlook/Hotmail: {outlook} | Yahoo: {yahoo} | Outros: {outros}")  # Exibe o total de cada provedor

```

```python
# Objetivo: sinalizar cargos proibidos
registros = [
    {"nome": "Paula", "cargo": "Assistente de Compras"},
    {"nome": "Rogério", "cargo": "Gerente"},
    {"nome": "Iasmin", "cargo": "Estagiario"},
    {"nome": "Bruno", "cargo": "Analista Sênior"}
]

proibidos = ["estagiario", "temporario"]

for r in registros:
    cargo = r["cargo"].strip().lower()
    bloqueado = False
    for termo in proibidos:
        if termo in cargo:
            bloqueado = True
            break
    print(
        f"{'BLOQUEIO' if bloqueado else 'OK'}: {r['nome']}"
        f"{' - Cargo não autorizado para aprovar pedidos.' if bloqueado else ' - Pode aprovar.'}
        "
    )
```

```python
# Objetivo: sinalizar cargos proibidos
registros = [  # Cria uma lista de dicionários, cada um contendo o nome e o cargo de uma pessoa
    {"nome": "Paula", "cargo": "Assistente de Compras"},
    {"nome": "Rogério", "cargo": "Gerente"},
    {"nome": "Iasmin", "cargo": "Estagiario"},
    {"nome": "Bruno", "cargo": "Analista Sênior"},
]

proibidos = ["estagiario", "temporario"]  # Define as palavras que indicam cargos proibidos

for r in registros:  # Percorre cada dicionário da lista 'registros'
    cargo = r["cargo"].strip().lower()  # Pega o valor do cargo, tira espaços das pontas e converte tudo para minúsculas
    bloqueado = False  # Começa assumindo que o cargo não é bloqueado
    for termo in proibidos:  # Percorre cada termo da lista de proibidos
        if termo in cargo:  # Verifica se o termo proibido aparece dentro do texto do cargo
            bloqueado = True  # Se sim, marca como bloqueado
            break  # Interrompe a verificação (não precisa continuar)
    print(  # Exibe o resultado da verificação
        f"{'BLOQUEIO' if bloqueado else 'OK'}: {r['nome']}"  # Mostra “BLOQUEIO” ou “OK” seguido do nome da pessoa
        f"{' - Cargo não autorizado para aprovar pedidos.' if bloqueado else ' - Pode aprovar.'}"  # Mostra o motivo ou permissão
    )
```

```python
# Objetivo: gerar "slug" simples sem acentos (didático)
itens = [
    "Câmera Profissional 4K",
    "Monitor 27'' UltraWide",
    "Cabo USB-C – Carregamento Rápido",
    "Impressora Jato de Tinta"
]

for titulo in itens:
    t = titulo.strip().lower()
    t = (t.replace("á","a").replace("à","a").replace("â","a").replace("ã","a")
           .replace("é","e").replace("ê","e").replace("í","i")
           .replace("ó","o").replace("ô","o").replace("õ","o")
           .replace("ú","u").replace("ç","c"))
    t = (t.replace("  ", " ")
           .replace(" ", "-")
           .replace("--", "-")
           .replace("’", "").replace("'", "")
           .replace("–", "-"))
    print(f"Título: {titulo} | SKU/slug: {t}")

```

```python
# Objetivo: gerar "slug" simples sem acentos (didático)
itens = [  # Cria uma lista de nomes de produtos com acentos e símbolos
    "Câmera Profissional 4K",
    "Monitor 27'' UltraWide",
    "Cabo USB-C – Carregamento Rápido",
    "Impressora Jato de Tinta"
]

for titulo in itens:  # Percorre cada item da lista
    t = titulo.strip().lower()  # Remove espaços das pontas e deixa tudo em minúsculas
    t = (  # Substitui manualmente os caracteres acentuados por suas versões simples
        t.replace("á", "a").replace("à", "a").replace("â", "a").replace("ã", "a")
         .replace("é", "e").replace("ê", "e")
         .replace("í", "i")
         .replace("ó", "o").replace("ô", "o").replace("õ", "o")
         .replace("ú", "u").replace("ç", "c")
    )
    t = (  # Ajusta o texto para o formato de slug, trocando espaços e símbolos
        t.replace("  ", " ")  # Substitui espaços duplos por um único espaço
         .replace(" ", "-")  # Troca espaços por hífens
         .replace("--", "-")  # Evita hífens duplicados
         .replace("'", "")  # Remove apóstrofos
         .replace('"', "")  # Remove aspas
         .replace("/", "-")  # Substitui barras por hífens
         .replace("–", "-")  # Substitui travessões por hífens
    )
    print(f"Título: {titulo} | SKU/slug: {t}")  # Exibe o nome original e o slug final gerado

```

Produto	Cor	Tamanho	SKU
Camiseta Básica	Preta	M	**CAM-PRE-M**
Camiseta Básica	Preta	G	**CAM-PRE-G**
Camiseta Básica	Branca	M	**CAM-BRA-M**

Significado de SKU

SKU vem do inglês ***Stock Keeping Unit***, que significa literalmente ***Unidade de Manutenção de Estoque***. É um código único que identifica cada produto dentro de um sistema de controle de estoque.

Pense no SKU como uma identificação exclusiva para cada item vendido ou armazenado, usada internamente pela empresa. Mesmo dois produtos muito parecidos podem ter ***SKUs*** diferentes se houver alguma diferença relevante – por exemplo, cor, tamanho ou versão.

Cada combinação gera um **SKU distinto**, que facilita:

- o **rastreamento** de produtos no sistema,
- a **reposição** de estoque,
- o **controle de vendas** e
- a **integração com sistemas de ERP, e-commerce e logística.**

Suponha que o título original seja:
"Câmera Profissional 4K"

O slug gerado seria:
"camera-profissional-4k"

Veja que:
- ✓ as letras foram convertidas para minúsculas,
- ✓ os acentos foram removidos,
- ✓ e os espaços foram trocados por hífens (-).

Assim, o nome fica mais limpo, padronizado e compatível com sistemas e URLs.

O que é um Slug

Um ***slug*** é uma forma simplificada e "amigável" de representar um título de texto em uma URL, nome de arquivo ou identificador de produto, sem espaços, acentos ou caracteres especiais. Em outras palavras, o slug é uma versão "limpa" e padronizada de um texto, usada principalmente para:

- endereços de páginas (URLs),
- nomes de arquivos,
- ou como identificadores legíveis em sistemas.

```python
# Objetivo: (auditoria de pedidos) resumo por status e revisão manual para pendentes >= 1000
pedidos = [
    {"id": 101, "cliente": "A1", "valor": 800.0,  "status": "pendente"},
    {"id": 102, "cliente": "A2", "valor": 1500.0, "status": "pendente"},
    {"id": 103, "cliente": "B7", "valor": 420.0,  "status": "aprovado"},
    {"id": 104, "cliente": "C2", "valor": 1200.0, "status": "aprovado"},
    {"id": 105, "cliente": "Z0", "valor": 50.0,   "status": "cancelado"}
]

aprovados = pendentes = cancelados = revisao_manual = 0
soma_aprovados = 0.0

for p in pedidos:
    status = p["status"].strip().lower()
    valor = float(p["valor"])
    if status == "aprovado":
        aprovados += 1
        soma_aprovados += valor
    elif status == "pendente":
        pendentes += 1
        if valor >= 1000.0:
            revisao_manual += 1
    elif status == "cancelado":
        cancelados += 1
    else:
        print(f"Status desconhecido no pedido: {p['id']}")

print(f"Aprovados: {aprovados} | Pendentes: {pendentes} (Revisão: {revisao_manual}) | Cancelados: {cancelados}")
print(f"Soma de aprovados: {soma_aprovados}")

```

```python
# Objetivo: (reconciliação leve de estoque) sinalizar itens abaixo do mínimo e fora de catálogo
estoque = [
    {"sku": "mouse-optico", "qtd": 12, "minimo": 10},
    {"sku": "teclado-mecanico", "qtd": 4, "minimo": 8},
    {"sku": "cabo-usb-c", "qtd": 0, "minimo": 5},
    {"sku": "monitor-27-ultrawide", "qtd": 2, "minimo": 2},
    {"sku": "fone-bluetooth", "qtd": 7, "minimo": 6}
]
catalogo_oficial = ["mouse-optico", "teclado-mecanico", "cabo-usb-c", "monitor-27-ultrawide"]

for item in estoque:
    sku, qtd, minimo = item["sku"], int(item["qtd"]), int(item["minimo"])
    if sku not in catalogo_oficial:
        print(f"ALERTA: {sku} - no estoque, mas fora do catálogo oficial.")
        continue
    print(f"{'REPOR' if qtd < minimo else 'OK'}: {sku} - qtd: {qtd} | mínimo: {minimo}")

```

Cadência de códigos — range() com for

Contagens, janelas, lotes, índices e número exato de iterações.

```python
# etiquetas 1..N
n = 5
for numero in range(1, n + 1):
    print(f"Etiqueta: {numero}")
```

PROBLEMAS SAÍDA CONSOLE DE DEPURAÇÃO TERMINAL PORTAS

```
PS C:\Users\#_Ohana_#\Documents\Projeto_Git> & C:/Users/
thon/Python313/python.exe c:/Users/#_Ohana_#/Documents/P
Etiqueta: 1
Etiqueta: 2
Etiqueta: 3
Etiqueta: 4
Etiqueta: 5
PS C:\Users\#_Ohana_#\Documents\Projeto_Git>
```

```python
# janela de horário comercial
for hora in range(9, 18):  # 9..17
    print(f"Agendar lembrete às {hora}h")

```

PROBLEMAS SAÍDA CONSOLE DE DEPURAÇÃO TERMINAL PORTAS

```
PS C:\Users\#_Ohana_#\Documents\Projeto_Git> & C:/Users/#_O
thon/Python313/python.exe c:/Users/#_Ohana_#/Documents/Proj
Agendar lembrete às 9h
Agendar lembrete às 10h
Agendar lembrete às 11h
Agendar lembrete às 12h
Agendar lembrete às 13h
Agendar lembrete às 14h
Agendar lembrete às 15h
Agendar lembrete às 16h
Agendar lembrete às 17h
PS C:\Users\#_Ohana_#\Documents\Projeto_Git>
```

```python
# mascarar CPF (últimos 3)
cpf = "12345678901"
mascarado = ""
for i in range(len(cpf)):
    mascarado += "*" if i < len(cpf) - 3 else cpf[i]
print(f"CPF mascarado: {mascarado}")

```

PROBLEMAS SAÍDA CONSOLE DE DEPURAÇÃO TERMINAL PORTAS

```
PS C:\Users\#_Ohana_#\Documents\Projeto_Git> & C:/Users/#_Ohana_#/App
thon/Python313/python.exe c:/Users/#_Ohana_#/Documents/Projeto_Git/py
CPF mascarado: *********901
PS C:\Users\#_Ohana_#\Documents\Projeto_Git>
```

```python
# paginação de 3 em 3
registros = ["r1", "r2", "r3", "r4", "r5", "r6", "r7"]
tamanho = 3
total = len(registros)

for inicio in range(0, total, tamanho):
    pagina = registros[inicio:inicio+tamanho]
    print(f"Página: {pagina}")
```

PROBLEMAS SAÍDA CONSOLE DE DEPURAÇÃO TERMINAL PORTAS

```
PS C:\Users\#_Ohana_#\Documents\Projeto_Git> & C:/Users/#_Ohana_#/AppD
thon/Python313/python.exe c:/Users/#_Ohana_#/Documents/Projeto_Git/pyt
Página: ['r1', 'r2', 'r3']
Página: ['r4', 'r5', 'r6']
Página: ['r7']
PS C:\Users\#_Ohana_#\Documents\Projeto_Git>
```

```python
# processamento em lotes
catalogo = ["  Produto A  ", "PRODUTO B", "produto C  ",
            "  Produto D", " Produto E", "produto F   ",
            "gadget X ", "   gadget Y"]
lote = 4
total = len(catalogo)

for inicio in range(0, total, lote):
    bloco = catalogo[inicio:inicio+lote]
    print(f"Processando bloco: {inicio} a {min(inicio+lote, total)-1}")
    for item in bloco:
        print(f" - Normalizado: {item.strip().title()}")

```

```python
# tentativas com backoff (refere-se a uma estratégia de espera progressiva entre tentativas de uma operação falhada para evitar sobrecarregar o sistema e dar tempo para que a condição que causou a falha seja resolvida.)
max_tentativas = 3
sucesso = False

for tentativa in range(1, max_tentativas + 1):
    print(f"Tentativa {tentativa} de {max_tentativas}")
    sucesso = (tentativa == 2)  # simulação
    if sucesso:
        print("Operação concluída com sucesso.")
        break
    print(f"Falhou, aguardando {'.' * tentativa}")

if not sucesso:
    print("Operação falhou após todas as tentativas.")

```

Combinação for + range() em cenários de mercado

```python
# comparação por índice (preços)
precos_antigos = [100, 250, 80, 199, 500]
precos_novos   = [110, 260, 88, 240, 550]

for i in range(len(precos_antigos)):
    antigo, novo = float(precos_antigos[i]), float(precos_novos[i])
    if novo <= antigo:
        print(f"OK idx {i} - sem aumento ou caiu: {antigo} -> {novo}")
        #"idx" é uma abreviação comum para "índice"
    else:
        variacao = (novo - antigo) / antigo
        if variacao > 0.10:
            print(f"ALERTA idx {i} - aumento > 10%: {antigo} -> {novo}")
        else:
            print(f"Aumento controlado idx {i}: {antigo} -> {novo}")

```

```python
# colunas de códigos
codigos = ["BR123", "BR456", "BR789", "BR101", "BR102", "BR103", "BR104"]
colunas = 3

for i in range(0, len(codigos), colunas):
    linha = codigos[i:i+colunas]
    print(" | ".join(linha))

```

PROBLEMAS SAÍDA CONSOLE DE DEPURAÇÃO TERMINAL PORTAS Python: python

```
PS C:\Users\#_Ohana_#\Documents\Projeto_Git> & C:/Users/#_Ohana_#/AppData/Local/Prog
thon/Python313/python.exe c:/Users/#_Ohana_#/Documents/Projeto_Git/python.py
BR123 | BR456 | BR789
BR101 | BR102 | BR103
BR104
PS C:\Users\#_Ohana_#\Documents\Projeto_Git>
```

```python
# média móvel (janela 3)
vendas = [100, 120, 150, 90, 130, 160, 170]
janela = 3
limite = len(vendas) - janela + 1

for i in range(limite):
    soma = 0
    for j in range(i, i + janela):
        soma += vendas[j]
    media = soma / janela
    print(f"Média móvel (dias {i+1} a {i+janela}): {media}")

```

```python
# verificador simples
series = ["123450", "987654", "111222", "432186"]

for s in series:
    s_limpo = s.strip()
    soma_pares = 0
    for i in range(len(s_limpo)):
        if i % 2 == 0:  # posições 0,2,4...
            soma_pares += int(s_limpo[i])
    print(f"{'OK' if soma_pares % 3 == 0 else 'FALHA'}: {s_limpo} (soma pares = {soma_pares})")

```

```python
# faixas de valor
valores = [12, 88, 100, 345, 500, 760, 250, 99, 499, 501, 1000, 15, 480]
faixa_0_99 = faixa_100_499 = faixa_500_up = 0

for i in range(len(valores)):
    v = float(valores[i])
    if v < 100:
        faixa_0_99 += 1
    elif v < 500:
        faixa_100_499 += 1
    else:
        faixa_500_up += 1

print(f"0..99: {faixa_0_99} | 100..499: {faixa_100_499} | 500+: {faixa_500_up}")

```

range() precisa do for?

- range() **define** a sequência; **não executa nada sozinho.**
- Normalmente, usamos **for + range()** para **percorrer** e **fazer algo** com cada número.

Exemplo incorreto (não imprime 0..4):

```
range(5)  # exibe "range(0, 5)" no terminal, não a contagem
```

Exemplo correto:

```
for i in range(5):
    print(i)
```

Aspecto	for em iteráveis (sem range)	for + range()	while
Ideia	“para cada item em X”	repetir **N** vezes / por índice	“enquanto condição for verdadeira”
Término	natural (fim do iterável)	natural (fim do range)	manual (você altera a condição)
Risco de loop infinito	baixo	baixo	maior
Quando usar	percorrer coleções	contagens/janelas/lotes/índices	interações abertas (menus, validação, até “sair”)

for/range() × while comparativo

- Use for para percorrer coleções e aplicar transformações/checagens item a item.
- Use for + range() para contagens, índices, janelas e lotes (lembre do fim exclusivo).
- Use while quando a repetição depende de uma condição aberta (ex.: até o usuário digitar “sair”).

```python
# identificação de entradas removidas e adicionadas.
base_antiga = ["A001", "A002", "A003", "A007", "A010"]
base_nova   = ["A001", "A003", "A004", "A010", "A011"]

sumiram, entraram = [], []

# Quem sumiu?
for i in range(len(base_antiga)):
    codigo = base_antiga[i]
    presente = False
    for j in range(len(base_nova)):
        if codigo == base_nova[j]:
            presente = True
            break
    if not presente:
        sumiram.append(codigo)

# Quem entrou?
for i in range(len(base_nova)):
    codigo = base_nova[i]
    presente = False
    for j in range(len(base_antiga)):
        if codigo == base_antiga[j]:
            presente = True
            break
    if not presente:
        entraram.append(codigo)

print(f"Sumiram: {sumiram}")
print(f"Entraram: {entraram}")
```

```
PS C:\Users\#_Ohana_#\Documents\Projeto_Git> 
thon/Python313/python.exe c:/Users/#_Ohana_#/
Sumiram: ['A002', 'A007']
Entraram: ['A004', 'A011']
PS C:\Users\#_Ohana_#\Documents\Projeto_Git>
```