

AED

- Aula 01 -

R E V I S A O D E

F U N D A M E N T O S

Conteúdo

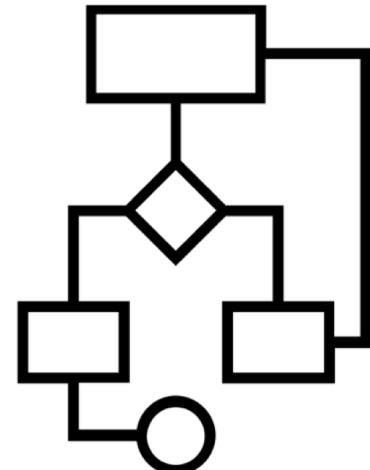
- O que são algoritmos
- Como construir algoritmos
- Alguns conceitos fundamentais
 - Variáveis
 - Comandos de atribuição
 - Comandos de entrada
 - Comandos de saída
- Estruturas Condicionais
- Estruturas de Repetição
- Exercícios

ALGORITMOS

Sequência lógica de instrução que podem ser executadas para a resolução de um problema

ALGORITMOS

- Ordenado
- Finito
- Não ambíguo



ALGORITMOS

Problema: Atravessar a rua



AtravessarRua

Olhar p/ direita

Olhar p/ esquerda

Se estiver vindo carro

Não atravesse

Senão

Atravesse

Fim AtravessarRua

ALGORITMOS

Problema: Preciso lavar o *jeans*, mas o bolso tá cheio de moedas



EsvaziarBolso

Enquanto Bolso não vazio

 Retira uma moeda

 Fim Enquanto

Fim EsvaziarBolso

Problema: Enquanto dirigia, percebi que um dos pneus furou



TrocarPneu

Identificar que o pneu furou

Pegar Ferramentas

...

Fim TrocarPneu

ALGORITMOS

TrocarPneu

- Identificar que o pneu furou
- Pegar Ferramentas e Pneu sobressalente
- Desapertar parafusos a meio termo
- Colocar Macaco e suspende o carro
- Desapertar totalmente os parafusos
- Tirar o Pneu furado
- Colocar o Pneu sobressalente
- Apertar parafusos
- Descer o carro
- Apertar melhor os parafuso
- Guardar ferramentas
- Reparar pneu furado (levar p/ a borracharia)

Fim TrocarPneu

Ambiguidade: Qual tipo de macaco?



ALGORITMOS

MeuJeitoDeTrocarnPneu

Identificar que o pneu furou

Se Marido tá no carro

TrocarnPneu(Marido)

Senão

Se Marido não tá viajando

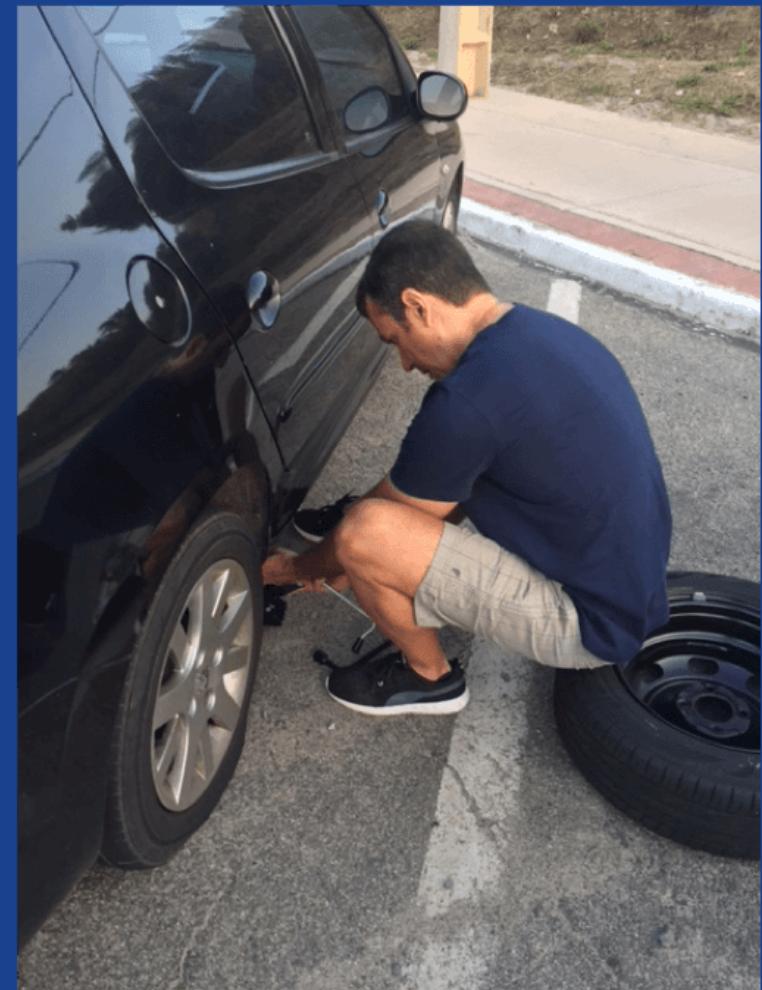
Liga p/ Marido

TrocarnPneu(Marido)

Senão

TrocarnPneu(Borracheiro)

Fim MeuJeitoDeTrocarnPneu



ALGORITMOS



Ingredientes

- 150g de margarina
- 4 ovos
- 1 ½ de xícara de açúcar
- ½ lata de leite condensado
- 200 ml leite de coco
- 2 xícaras de trigo c/fermento
- 100g de coco ralado

FazerBolo

Separe os ingredientes

Pré-aqueça o forno a 180°C

Unta e esfarinha uma forma

Na batedeira, bata a margarina, os ovos e o xícara de açúcar até forma uma pasta homegenea

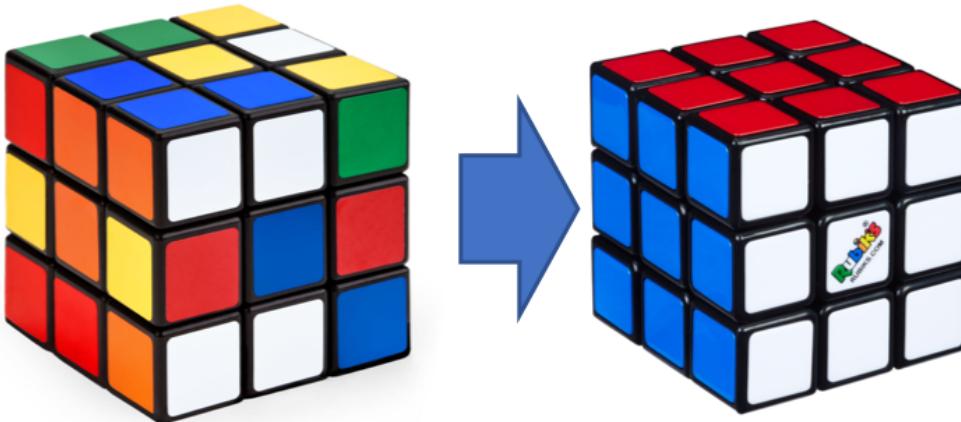
Misture o trigo, o leite condensado, o leite de coco e o coco ralado e bate novamente

Despeje a mistura na forma

Leve ao forno por 25 - 30 min

Fim FazerBolo

ALGORITMOS



ResolverRubiksCube

Completar a primeira camada

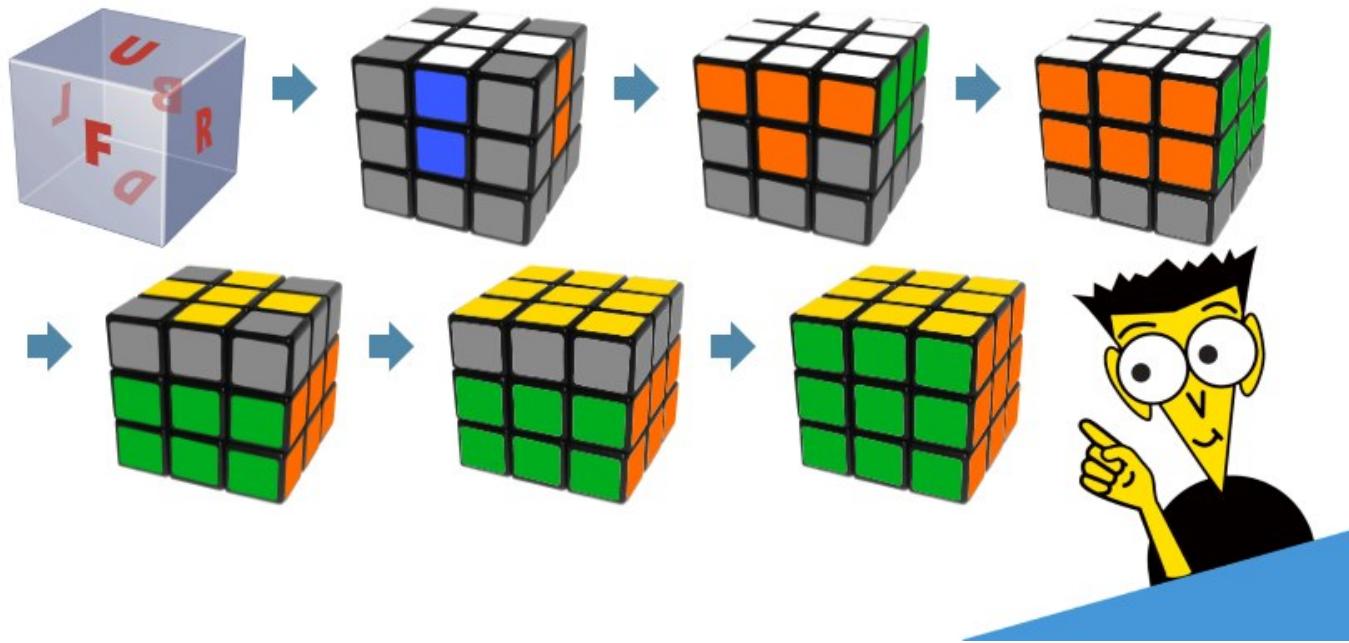
Completar a segunda camada

Completar a terceira Camada

Fim ResolverRubiksCube

ALGORITMOS

How to Solve The Rubik's Cube



Como construir algoritmos?

Algoritmos: Como fazer?

1. Entender o problema
2. 'Bolar' um plano para resolver o problema
3. Executar o Plano
4. Avaliar a solução

Três fases fundamentais



Exemplo 1

Escreva um algoritmo que leia um valor em real e mostre o valor em dólar.

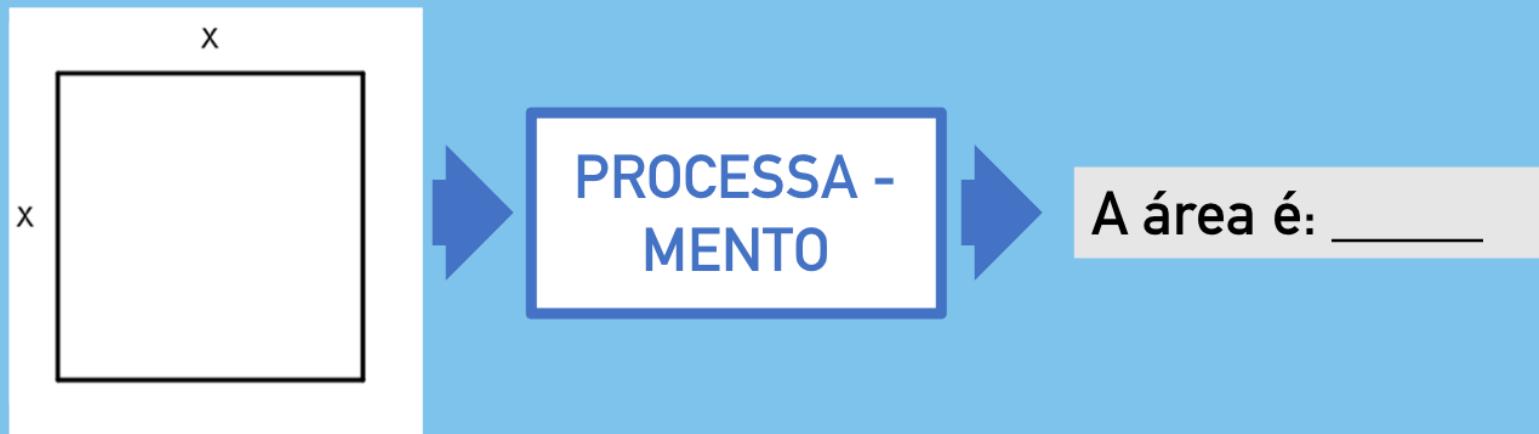


PROCESSA -
MENTO



Exemplo 2

Escreva um algoritmo que calcule a área de um quadrado.



Conceitos fundamentais: Variáveis

Variáveis

É um local nomeado na memória onde um programador pode armazenar dados.

- e, posteriormente, recuperar dados usando a variável pelo nome.
- O valor varia ao longo do tempo (programa), mas só pode armazenar um valor a cada instante
- Possui nome e tipo (Numérica, Textual ou Boleana)

Variáveis

Regras de nomes

- Deve começar com uma letra ou _
- Pode ser constituído de letras, números e sublinha (underscore).
- Distingue maiúsculo de minúsculo.

spam
_speed
eggs



permitidas

23spam
#speed
eggs.12



inaceitáveis

spam
Spam
SPAM



diferentes

Tipos de Variáveis

Variáveis Numéricas

Armazenam números. Podem ainda classificadas como inteiros, reais (float) ou complexas:

```
a = 2019  
print(type(a))
```

```
a = 1.5  
print(type(a))
```

```
a = 4+3j  
print(type(a))
```

Tipos de Variáveis

Variáveis Textuais (String)

Específicas para dados que contenham letras, números e espaços:

```
a = 'Este é 1 exemplo de string'  
print(type(a))
```

```
a = "Também pode ser usado aspas duplas"  
print(type(a))
```

Tipos de Variáveis

Variáveis Booleanas

Armazenam somente dados lógicos:
Verdadeiro ou Falso

```
a = True  
print(type(a))
```

```
a = False  
print(type(a))
```

Conceitos fundamentais: Comando de Atribuição

Comando de Atribuição

O comando de atribuição cria novas variáveis e dá valores a elas:

```
>>> mensagem = "E aí, Tudo bem?"  
>>> n = 17  
>>> pi = 3.14159
```

Conceitos fundamentais: Comando de Entrada

Comando de Entrada

INPUT

Para permitir flexibilidade, podemos querer receber a entrada do usuário. Em Python, temos a função `input()` para permitir isso.

```
# coding=utf-8
nome = input('Digite o seu nome: \t')
```

Conceitos fundamentais: Comando de Saída

Comando de Saída

PRINT

Usamos a função print () para enviar dados para o dispositivo de saída padrão (tela).

```
# coding=utf-8
disc = 'Algoritmos e Estrutura de Dados'
semestre = '2019.2'
print('Bem vindos à disciplina {} - {}'.format(disc, semestre))
```

Formatando a saída c/ print()

Separar por vírgula

```
x, y, z = 10, 20, 30  
print('altura=', x, 'largura=', y, 'profundidade=', z)
```

Marcadores

```
print('Hoje é %d de %s de %d.' % (27, 'agosto', 2019) )  
print('Hoje é %5d de %s de %d.' % (27,'agosto', 2019) )
```

Formatando a saída c/ print()

Usando Parâmetros

```
# coding=utf-8
disc = 'Algoritmos e Estrutura de Dados'
semestre = '2019.2'
print('Bem vindos à disciplina {} - {}'.format(disc, semestre))
```

Usando Parâmetros nomeados

```
# coding=utf-8
print('Olá, {nome} {sobrenome}'.format(nome = 'John', sobrenome = 'Oliveira'))
```

Estruturas Condicionais

Utilizadas para desvio do fluxo do programa

- Condicional Simples (if)
- Condicional Composta(if - else)
- Condicional encadeada
- Condicional aninhada (if-elif-else)
- Exercícios

Estruturas Condicionais

Operadores Relacionais

Operadores Booleanos

>, >=, <, <=, ==, !=

not, and, or

```
# coding=utf-8
idade = float(input('Digite a sua idade'))

maior_de_idade = idade >= 18

if not maior_de_idade:
    print('Acesso proibido')

else:
    print('Acesso liberado')
```

Condicional Simples

Se um determinado teste for verdadeiro, uma ação será executada.

Sintaxe

```
if (teste):  
    bloco_de_comandos
```

```
# coding=utf-8  
n1=float(input('Nota 1:'))  
n2=float(input('Nota 2:'))  
n3=float(input('Nota 3:'))  
media=(n1+n2+n3)/3  
  
if(media>=7):      print('aprovado')
```

Condicional Composta

Se um determinado teste for verdadeiro, uma ação será executada. Caso contrário, outra ação será executada.

Sintaxe

```
if (teste):  
    bloco_de_comandos  
else:  
    bloco_de_comandos
```

```
...  
media=(n1+n2+n3)/3  
if(media>=7):  
    print('aprovado')  
else:  
    print('reprovado')
```

Condicional Encadeada

```
if(media>=7):
    print('aprovado')
    print('Só alegria!')

elif(media>=4):
    print('Prova final')
    print('Vou estudar mais')

else:
    print('Reprovado')
    print('Vou estudar MUITO mais')
```

Estruturas de Repetição

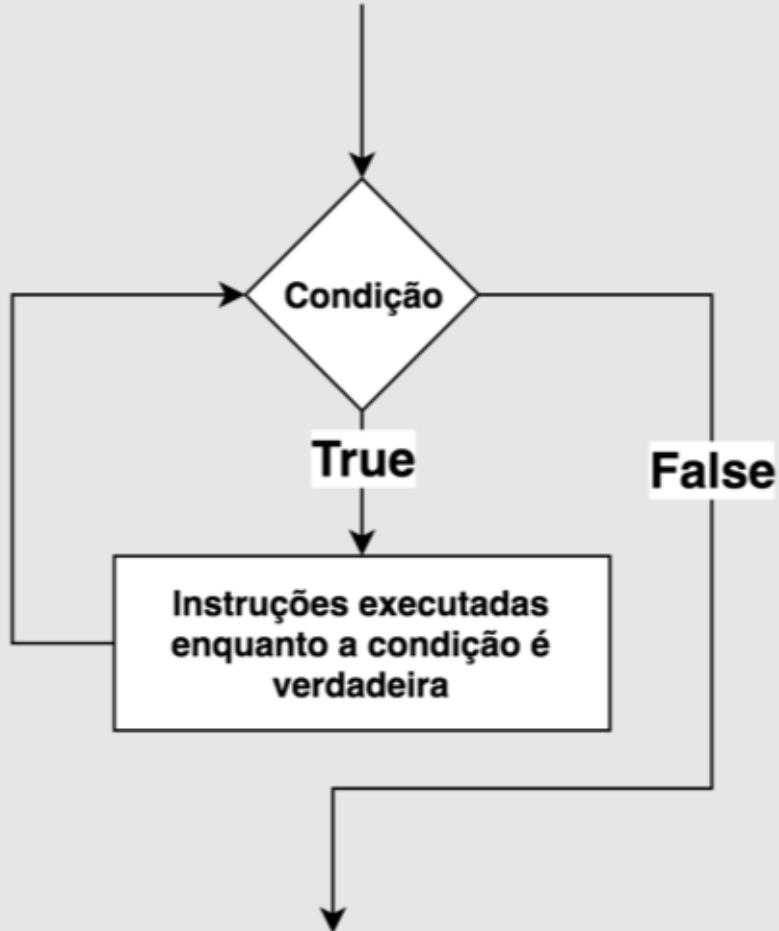
Em muitos algoritmos, faz-se necessário executar uma sequência de comandos repetidamente, em um laço (loop)

Python fornece os seguintes comandos de repetição:

while e for

Estruturas de Repetição: **WHILE**

while



Exemplo

```
x=1  
while x<=3:  
    print('x = ', x)  
    x = x + 1
```

Saída

```
x = 1  
x = 2  
x = 3
```

Comando while

O exemplo a seguir calcula a soma dos números de 1 a 100.

Observe o uso das variáveis i e n para controlar o número de repetições do loop

Exemplo

```
n = 100
soma = 0
i = 1
while (i <= n):
    soma = soma + i
    i += 1
print("Soma de 1 ate" , n, ":" , soma)
```

Saída

```
Soma de 1 até 100: 5050
```

Comando while

Exemplo

```
x = 0
while (x < 5):
    print(x)
    x += 1
```

Saída

```
0
1
2
3
4
```

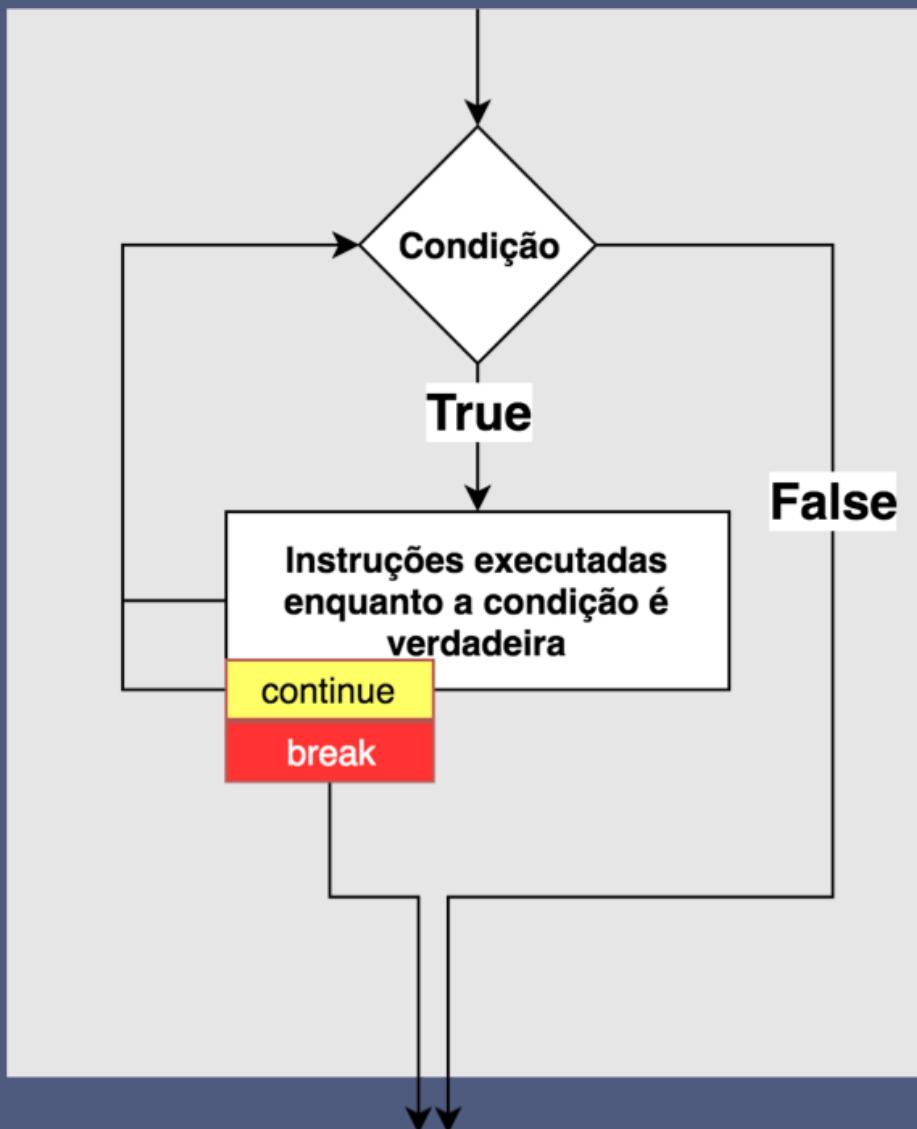
Exemplo

```
while 1:
    print("Loop infinito")
```

Saída

```
Loop infinito
Loop infinito
Loop infinito
Loop infinito
...
...
```

while com break



Exemplo

```
x=1
while x<=3:
    print('x = ', x)
    break
    x = x + 1
```

Saída

```
x = 1
```


while com break

Às vezes, deseja-se sair do loop imediatamente, quebrando a iteração atual. Para tanto, use o comando `break`.

Exemplo

```
x=1
while x<=10:
    if (i==5):
        break
    print('x =', x)
    x = x + 1
print('o valor de x é', x)
```

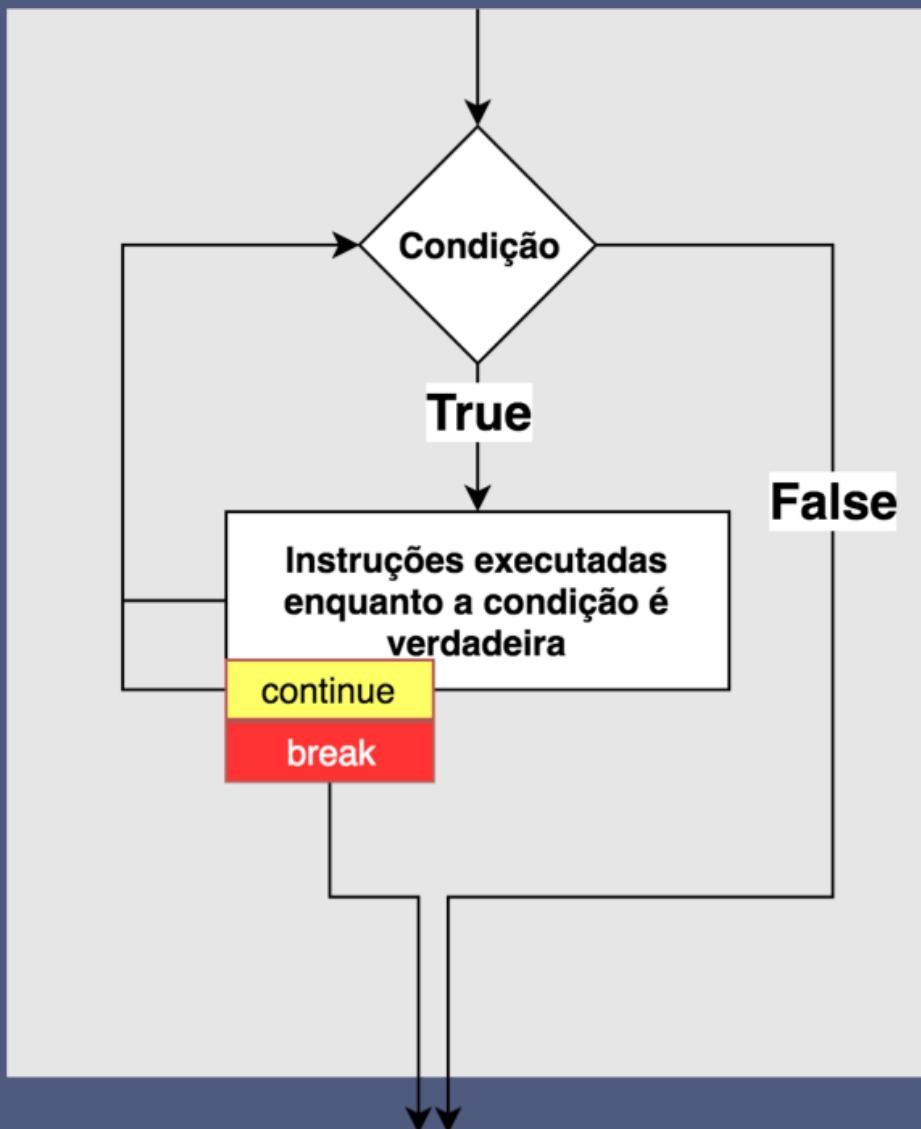
Saída

```
1
2
3
4
o valor de x é 5
```

Exercício

Escreva um programa que escolha aleatoriamente um numero entre 1 e 10, ao qual o usuário deve tentar adivinhar. O jogo acaba quando o usuário acerta ou quando ele desiste (digitando 0).

while com continue



Exemplo

```
x=1
while x<=3:
    print('x = ', x)
    continue
    x = x + 1
```

Saída

```
x = 1
x = 1
x = 1
x = 1
....
```


while com continue

O comando `continue` promove a interrupção da iteração atual do loop, passando para a seguinte.

Exemplo

```
i = 0
while (i <= 5):
    i += 1
    if (i==3):
        continue
    print(x)
```

Saída

```
1
2
4
5
```

Quando usar o comando **WHILE?**

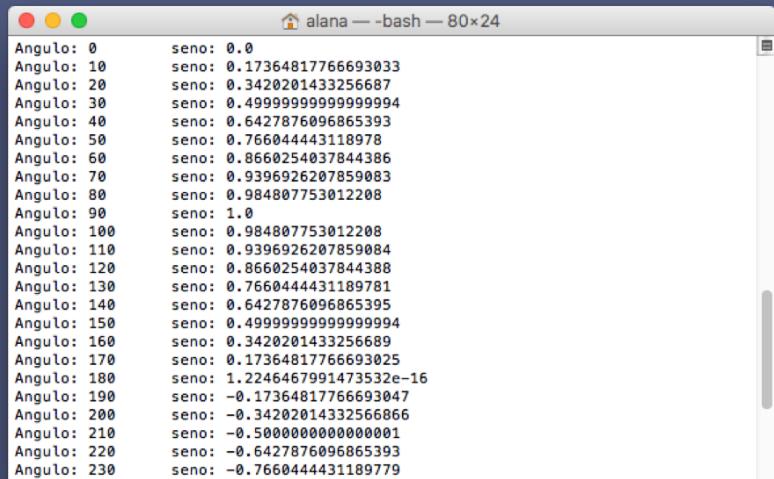
O while é adequado quando não há como determinar quantas iterações vão ocorrer.

Por exemplo: Escreva um algoritmo que leia o código do item pedido, a quantidade e calcule o valor a ser pago por aquele lanche. O algoritmo só finaliza se o código informado por -999.

Exercício

Escreva um programa que calcule e mostre o seno dos ângulos entre 0 e 360 (de 10 em 10).

dica: Use a função sin do pacote math



```
alana — bash — 80x24
Angulo: 0      seno: 0.0
Angulo: 10     seno: 0.17364817766693033
Angulo: 20     seno: 0.3420201433256687
Angulo: 30     seno: 0.49999999999999994
Angulo: 40     seno: 0.6427876096865393
Angulo: 50     seno: 0.76604443118978
Angulo: 60     seno: 0.8660254037844386
Angulo: 70     seno: 0.9396926207859083
Angulo: 80     seno: 0.984807753012208
Angulo: 90     seno: 1.0
Angulo: 100    seno: 0.984807753012208
Angulo: 110    seno: 0.9396926207859084
Angulo: 120    seno: 0.8660254037844388
Angulo: 130    seno: 0.766044431189781
Angulo: 140    seno: 0.6427876096865395
Angulo: 150    seno: 0.49999999999999994
Angulo: 160    seno: 0.3420201433256689
Angulo: 170    seno: 0.17364817766693025
Angulo: 180    seno: 1.2246467991473532e-16
Angulo: 190    seno: -0.17364817766693047
Angulo: 200    seno: -0.34202014332566866
Angulo: 210    seno: -0.5000000000000001
Angulo: 220    seno: -0.6427876096865393
Angulo: 230    seno: -0.766044431189779
```

Estruturas de Repetição: FOR

FOR

O loop for de Python é baseado em iteradores, ou seja, ele percorre uma sequência de itens e realiza as ações apropriadas sobre cada um deles

Os itens podem ser elementos de listas, tuplas, strings, chaves de dicionários e outros “iterables”

FOR

Exemplo

```
for item in ['1o', '2o', '3o']:  
    print(item)
```

O for é adequado quando pode-se determinar quantas iterações vão ocorrer ou há uma sequência a seguir.

Comando FOR

- O break interrompe completamente o laço.
- O continue passa para a próxima iteração.
- O else é executado ao final do laço, a não ser que o laço tenha sido interrompido por break.

Exercício

- Escreva um programa que some todos os números entre 0 e 99.

Exercício

- Escreva um programa que mostre todos os múltiplos de 5 entre 0 e 50.

Exercício

- Escreva um programa que mostre os dias da semana.

Repetições Aninhadas

Repetições Aninhadas

- É possível utilizar repetições dentro de outras repetições

*Exemplo: Tabela de multiplicação de
0 a 10 usando while.*

Repetição Aninhada

```
i = 0
while i<= 10:
    j = 0
    print('===== TABUADA DE ===== '.format(i))
    while j<=10:
        print('{ }x{ }={ }'.format(i,j,i*j))
        j=j+1
    i=i+1
```

Repetição Aninhada

===== TABUADA DE 0 =====

```
0x0=0  
0x1=0  
0x2=0  
0x3=0  
0x4=0  
0x5=0  
0x6=0  
0x7=0  
0x8=0  
0x9=0  
0x10=0
```

===== TABUADA DE 1 =====

```
1x0=0  
1x1=1  
1x2=2  
1x3=3  
1x4=4  
1x5=5  
1x6=6  
1x7=7  
1x8=8  
1x9=9  
1x10=10
```

===== TABUADA DE 2 =====

```
2x0=0  
2x1=2  
2x2=4  
2x3=6  
2x4=8
```

===== TABUADA DE 2 =====

```
2x0=0  
2x1=2  
2x2=4  
2x3=6  
2x4=8  
2x5=10  
2x6=12  
2x7=14  
2x8=16  
2x9=18  
2x10=20
```

===== TABUADA DE 3 =====

```
3x0=0  
3x1=3  
3x2=6  
3x3=9  
3x4=12  
3x5=15  
3x6=18  
3x7=21  
3x8=24  
3x9=27  
3x10=30
```

===== TABUADA DE 4 =====

```
4x0=0  
4x1=4  
4x2=8  
4x3=12  
4x4=16
```

===== TABUADA DE 9 =====

```
9x0=0  
9x1=9  
9x2=18  
9x3=27  
9x4=36  
9x5=45  
9x6=54  
9x7=63  
9x8=72  
9x9=81  
9x10=90
```

===== TABUADA DE 10 =====

```
10x0=0  
10x1=10  
10x2=20  
10x3=30  
10x4=40  
10x5=50  
10x6=60  
10x7=70  
10x8=80  
10x9=90  
10x10=100
```

Exercícios

01 - Desenvolver um algoritmo que leia a altura de 15 pessoas. Este programa deverá calcular e mostrar :

- a) A menor altura do grupo;*
- b) A maior altura do grupo;*

02 - Desenvolver um algoritmo que leia um número não determinado de valores e calcule e escreva a média aritmética dos valores lidos, a quantidade de valores positivos, a quantidade de valores negativos e o percentual de valores negativos e positivos

03 - Escrever um algoritmo que leia uma quantidade desconhecida de números e conte quantos deles estão nos seguintes intervalos: [0-25], [26-50], [51-75] e [76-100]. A entrada de dados deve terminar quando for lido um número negativo

04 - Faça um algoritmo estruturado que leia uma quantidade não determinada de números positivos. Calcule a quantidade de números pares e ímpares, a média de valores pares e a média geral dos números lidos. O número que encerrará a leitura será zero

05 - Escrever um algoritmo que gera e escreve os números ímpares entre 100 e 200.

06 - Escreva um algoritmo que leia um valor inicial A e uma razão R e imprima uma seqüência em P.A. contendo 10 valores.

07 - Escreva um algoritmo que leia um valor inicial A e uma razão R e imprima uma seqüência em P.G. contendo 10 valores.

08 - Escreva um algoritmo que leia um valor inicial A e imprima a seqüência de valores do cálculo de A! e o seu resultado. Ex: $5! = 5 \times 4 \times 3 \times 2 \times 1 = 120$

09 - O restaurante a quilo Bem-Bom cobra R\$52,90 por cada quilo de refeição. Escreva um algoritmo que leia o peso do prato montado pelo cliente (em quilos) e imprima o valor a pagar. Assuma que a balança já desconte o peso do prato.