Estruturas de Controle em Python

Introdução à Programação para Biologia Molecular

Rosane Minghim
Apoio na confecção: Danilo Medeiros Eler
Rogério Eduardo Garcia
Renato Rodrigues
Carlos E. A. Zampieri

Baseado na Apostila: Curso Introdutório de Computação por R. Minghim e G. P. Telles

Blocos de Comandos

- O uso de **tabulação** "espaços em branco a partir do início do bloco" é necessário para definir o bloco de comandos.

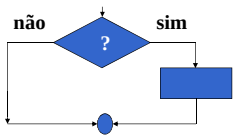
```
se condição então
    comandos
```

Tabulação

Escolha Simples

- Pode-se selecionar a sequência de comandos a ser executada
- Formato:

```
if condição :
    Comandos
```



Exemplo de Escolha Simples

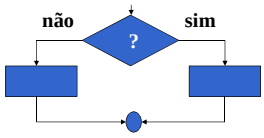
```
if (empregado == diarista):
    salario = salario_base + dias_trabalhados
              * valor_do_dia
```

```
if (empregado == diarista):
    diarista = diarista + 1
    adicional = dias_trabalhados * valor_do_dia
    salario = salario_base + adicional
```

Escolha Composta

- Formato:

```
if condição :
        Comandos
else :
        Comandos
```



Exemplo

```
if (empregado == diarista) :
    salario = salario_base +
          dias_trabalhados * valor_do_dia
else :
    complemento = fator *
                  complemento_basico
    salario = salario_base + complemento
```


Regras Básicas

- Usar sempre tabulação para definir os comandos a serem executados dentro da escolha.
- Atenção com as tabulações ao programar

```
if (empregado == diarista) :
    salario = salario_base + dias_trabalhados
                * valor_do_dia
```

Tabulação Correta

```
if (empregado == diarista) :
    salario = salario_base + dias_trabalhados
                * valor_do_dia
else :
    adicional = dias_trabalhados *
                valor_do_dia
salario = salario_base + adicional
```

Tabulação Errada

Regras Básicas

```
if (empregado == diarista):
   salario = salario_base +
             dias_trabalhados*valor_do_dia
else:
   complemento = fator*complemento_basico
   salario = salario_base + complemento
```

Exemplo

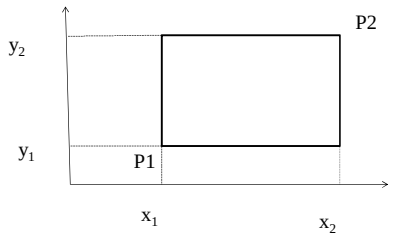
```
numero1 = input('Digite o numero 1')
numero2 = input('Digite o numero 2')
numero3 = input('Digite o numero 3')

if (numero1>numero2) :
   if (numero2>numero3) or (numero2 == numero3) :
      maior = numero1
   else :
      if (numero1>numero3) :
         maior = numero1
      else :
         maior = numero3
else :
   if (numero2 > numero3) or (numero2 == numero3) :
      maior = numero2
   else :
      maior = numero3

print maior
```


Relembrando a Prática I

Desenvolver um algoritmo que receba do usuário a informação de um tipo de figura geométrica (retângulo, círculo ou quadrado), identificada por um número inteiro. O algoritmo lerá os dados necessários para a figura e calculará e imprimirá seu perímetro e sua área. Para o retângulo e o quadrado o algoritmo lerá as coordenadas dos vértices opostos. Pode assumir que a figura possui lados paralelos aos eixos x e y. Para o círculo o algoritmo lerá o tamanho do raio. Editar o algoritmo num arquivo Figuras.txt.

Relembrando a Prática I

Desenvolver um algoritmo que receba do usuário a informação de um tipo de figura geométrica (retângulo, círculo ou quadrado), identificada por um número inteiro. O algoritmo lerá os dados necessários para a figura e calculará e imprimirá seu perímetro e sua área. Para o retângulo e o quadrado o algoritmo lerá as coordenadas dos vértices opostos. Pode assumir que a figura possui lados paralelos aos eixos x e y. Para o círculo o algoritmo lerá o tamanho do raio. Editar o algoritmo num arquivo Figuras.txt.




```
area <- abs(x2-x1)*abs(y2-y1)
perimetro <- 2*(abs(x2-x1)+abs(y2-y1))
```

```
se figura = RETANGULO então
      area <- abs(x2-x1) * abs(y2-y1)
      perimetro <- 2*(abs(x2-x1) + abs(y2-y1))
fim se
```

13

Relembrando a Prática I

Desenvolver um algoritmo que receba do usuário a informação de um tipo de figura geométrica (retângulo, círculo ou quadrado), identificada por um número inteiro. O algoritmo lerá os dados necessários para a figura e calculará e imprimirá seu perímetro e sua área. Para o retângulo e o quadrado o algoritmo lerá as coordenadas dos vértices opostos. Pode assumir que a figura possui lados paralelos aos eixos x e y. Para o círculo o algoritmo lerá o tamanho do raio.
Editar o algoritmo num arquivo Figuras.txt.

14

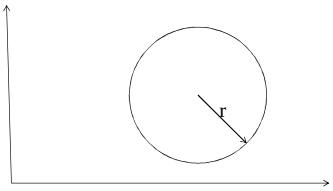
```
se figura = RETANGULO OU figura = QUADRADO então
      area <- abs(x2-x1) * abs(y2-y1)
      perimetro <- 2*(abs(x2-x1) + abs(y2-y1))
fim se
```

15

Relembrando a Prática I

Desenvolver um algoritmo que receba do usuário a informação de um tipo de figura geométrica (retângulo, círculo ou quadrado), identificada por um número inteiro. O algoritmo lerá os dados necessários para a figura e calculará e imprimirá seu perímetro e sua área. Para o retângulo e o quadrado o algoritmo lerá as coordenadas dos vértices opostos. Pode assumir que a figura possui lados paralelos aos eixos x e y. Para o círculo o algoritmo lerá o tamanho do raio.
Editar o algoritmo num arquivo Figuras.txt.

16




```
area <- PI * R  * R
perimetro <- 2* PI * R
```

17

```
se (figura = RETANGULO OU figura = QUADRADO)então
    leia (x1,y1,x2,y2)
    lado1 <- abs(x2-x1)
    lado2 <- abs(y2-y1)
    area <- lado1 * lado2
    perimetro = 2*(lado1 + lado2)
  senão
    se figura = CIRCULO então
      leia (r)
      area = PI*r*r
      perimetro = PI*r*2
    fim se
  fim se
```

18

Pseudo-Código Completo

Ver arquivo pratica1_algoritmo.txt

Código Python Completo

Ver arquivo pratica1_noloop.py

19

Preocupações:

1 - Max – sempre fazer o algoritmo antes

2 - organização do código
nomes apropriados de variáveis e constantes
definir, adotar e manter um padrão pra o nome de variáveis
inicializações devidas
comentários adequados

3 - diálogo com o usuário
mensagens adequadas de entrada e saída
formatação que facilite a leitura
cuidados com maiúsculas e minúsculas, além de outros caracteres ambíguos.

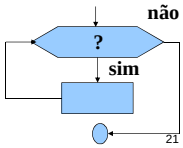
4 - checagem de erros

20

Repetição por Condição

- Uma das formas de repetir um conjunto de comandos de um algoritmo é subordiná-los a um comando de repetição usando uma estrutura da forma:

```
while condição :
   bloco de comandos
```



Exemplo

```
Algoritmo calcula_senos

 variável
   n, i: inteiro

 leia(n)
 i ← 0
 enquanto i ≤ n faça
   escreva(seno(i))
   i ← i + 1
 fim enquanto
fim
```

```
import math

n = input()

i = 0
while i <= n :
   #Converte angulo i para radianos
    angulo = i*math.PI/180
    print 'seno de ',i,'graus = ',
           math.sin(angulo)
    i = i + 1
```


Exemplos de laço (repetição) baseados na Prática I

Ver arquivos:

pratica1_loop_input.py

pratica1_loop_whole.py

pratica1_loop_looprange.py

Repetição por Contagem

- Na iteração baseada em contagem, sabe-se ***antecipadamente*** quantas vezes um conjunto de comandos vai ser repetido.
- Formato:

```
for var in range(valor_inicial,valor_final,valor_do_passo):
   bloco de comandos
```

Exemplo 1

```
n = input('Digite o final da contagem: ')

for i in range(1,n):
  print i,' ‘
```

25

Exemplo 2

```
print 'Conta decrescente\n’

n = input('Digite o valor do inicio da contagem: ')

for i in range(n,1,-1):
  print i,' ‘
```

26

Exemplo 3

```
n = input('Digite o final da contagem: ‘)

for i in range(1,n/2):
  print (i*2-1),' '
```

27

Mais exemplos

- Nos códigos-exemplo que acompanham esses slides
- Tarefa:
- Estudar, reproduzir e modificar os códigos exemplo que acompanham esses slides.
- **FIM DOS SLIDES**

28