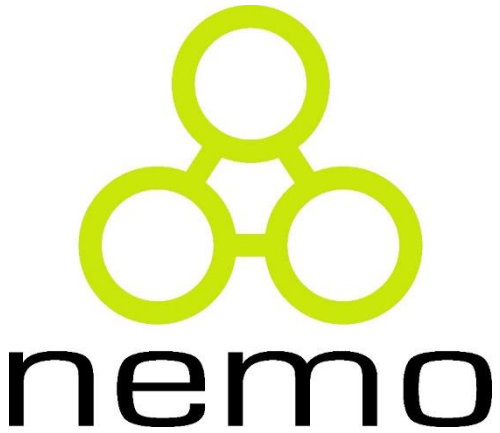


Programação I

Jordana S. Salamon

jssalamon@inf.ufes.br

jordanasalamon@gmail.com

DEPARTAMENTO DE INFORMÁTICA

CENTRO TECNOLÓGICO

UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO

A Arte de Resolver Problemas

REVISÃO

- A resolução de um problema como um processo complexo que se divide em quatro etapas
  - Compreensão do Problema
  - Planejamento
  - Desenvolvimento
  - Avaliação do processo e seus resultados

nemo

Etapa 1 - Compreensão do Problema

REVISÃO

- Perguntas tais como:
  - Quais são os dados de entrada?
  - O que desejamos produzir como resultado?
  - Qual a relação que existe entre os dados de entrada e o resultado a ser produzido?
  - Quais são as propriedades importantes que os dados de entrada possuem?

nemo

Etapa 2 – Planejamento (Estratégia)

REVISÃO

- Perguntas:
  - Qual a relação que existe entre este problema que temos e um outro problema já conhecido?
  - Será que podemos quebrar o problema em problemas menores?
  - Será que generalizando o problema não chegaremos a um outro já conhecido?
  - Conhecemos um problema parecido, embora mais simples, o qual quando generalizado se aproxima do que temos?

nemo

Etapa 3 – Desenvolvimento (Construção)

REVISÃO

- Nesta fase, devemos considerar os elementos da linguagem de programação que iremos usar, respeitando os elementos disponibilizados pela linguagem:
  - Tipos de dados
  - Formas de definição
  - Possibilidades de generalização
  - (re)Uso de elementos anteriormente definidos.

nemo

Etapa 3 - Desenvolvimento

REVISÃO

- Em síntese, a fase de desenvolvimento compreende as seguintes subfases:
  - construção da solução;
  - planejamento do teste;
  - execução manual do teste;
  - codificação da solução;
  - teste com o uso do computador.

nemo

Etapa 4 - Avaliação

REVISÃO

- Perguntas:
  - Este foi o melhor caminho que poderia ter sido usado?
  - Será que desdobrando a solução não obtenho componentes que poderei usar mais facilmente no futuro?
  - Se esta solução for generalizada é possível reusá-la mais facilmente em outras situações?
- Registre tudo, organize-se para a resolução de outros problemas. Anote suas decisões, enriqueça a sua biblioteca de soluções e métodos.

nemo

Tipos de Dados

- Denominamos Tipo de Dados a um conjunto de valores, munido de um conjunto de operações sobre esses valores.

- Por exemplo, podemos denominar de T1 ao tipo de dados formado por um conjunto S de valores idêntico aos números naturais (S = {0,1,2,3, ...}) e munido das operações de adição (a) e multiplicação (m).

- Cada operação, por sua vez, relaciona um (ou mais) conjunto(s) de valores do domínio a outro conjunto de valores do contradomínio.

nemo

Tipos de Dados

- Para o tipo T1, o domínio de ‘a’ é o produto cartesiano S X S e o contradomínio é S.

- A notação a seguir, semelhante à notação matemática, é usualmente utilizada e em geral é denominada de “assinatura” da operação.
- a :: S X S ▫ S
- m :: S X S ▫ S

- Esta notação significa que a operação de adição a opera dois valores do conjunto S e resulta em um valor do mesmo conjunto.

Tipos de Dados

- Usamos até agora os tipos Numéricos:
  - Tipo Inteiro: Representa um intervalo finito do conjunto matemático dos números inteiros (int e long).
  - Tipo Real: Representa um intervalo finito do conjunto matemático dos números reais (float).
    - IMPORTANTE: A parte inteira do número real é separada das casas decimais por ponto!

      >>> 3.2

    - Uma outra notação possível para números reais é a notação científica, denotada por uma base, seguida da letra „e" ou „E" e o expoente.

```
>>> 10e-5
0.0001
```

nemo

Tipos Numéricos

- Operadores Numéricos Básicos:
  - SOMA: +
  - SUBTRAÇÃO: -
  - MULTIPLICAÇÃO: *
  - DIVISÃO REAL: / >>> 1.0 / 2.0

    0.5
  - DIVISÃO INTEIRA: // (arredonda para baixo)

    >>> 1 // 2 >>> 1.0 // 2.0

    0 0.0
  - RESTO DA DIVISÃO INTEIRA: %
  - POTENCIAÇÃO: **

nemo

Tipos Numéricos

- Coerção numérica
  - Automática

```
>>> 1 + 4.5
5.5
```

  - Forçada

```
>>> float(3)
3.0
>>> int(3.7)
3
```

- Arredondamento

```
>>> round(3.75)
4.0
```

- Ademais, pode-se fazer a coerção forçada usando as seguintes funções:

Nome	Descrição	Exemplos
float(x)	Converte um número inteiro em um número real	`>>> float(3)` `3.0`
int(x)	Converte um número real em um número inteiro, eliminando a parte decimal	`>>> int(3.75)` `3`
round(x)	Arredonda um número real	`>>> round(3.75)` `4.0`

nemo

Tipo Caracter/String

- Letras, símbolos e até mesmo números representados entre aspas simples ou duplas

```
>>> def Ola():
        return “Ola Mundo!!!”
>>> Ola()
‘Ola Mundo!!'
```

Proposições Lógicas

- Afirmações sobre as quais podemos discutir quanto à veracidade:
  - Três é maior que dois.
  - Todo número primo é ímpar.
  - Hoje é domingo.

- **Sentenças Fechadas**: todos os componentes estão devidamente explicitados.

Proposições Lógicas

- **Sentenças Abertas**: alguns componentes não estão explicitados.
  - x + 5 < 10

- O valor verdade apenas pode ser avaliado quanto instanciamos, ou seja, quando atribuímos valores aos componentes em aberto:
  - 3 + 5 < 10
  - 6 + 5 < 10

Proposições Lógicas

- **Sentenças Compostas**: podemos combinar sentenças de forma que seus valores verdade componham um valor verdade final
  - Três é maior que dois **e** dois é maior que um.
  - Hoje é domingo **ou** hoje é segunda.
  - **Nem** todo número primo é ímpar.
  - $x + 5 < 10$ **e** $2x < 5$

- Usamos três **operadores lógicos** para compor as sentenças:
  - **e**, **ou** e **não** (nem)

Expressões lógicas em Python

- Operadores Lógicos

Operação lógica	Operador lógico (Python)
e	and
ou	or
não	not

- Operadores Relacionais

Operador	*Significado*	*Exemplo*	*Resultado*
==	igualdade	(2 + 3) == (8 – 3)	True
!=	Diferença	5 != (4 * 2 -3)	False
<	Menor	(2 + 3) < 6	True
<=	Menor ou igual	(2 * 3) <= 6	True
>	Maior	(4 + 2) > (2 * 3)	False
>=	Maior ou igual	(8 – 3 * 2) >= 15	False

nemo

Expressões lógicas em Python

```
>>> 5 > 4
True
>>> 4 != 4 + 1
True
>>> 5 % 2 == 1
True
>>> 3 > 2
True
>>> 3 < 2
False
```

Tipo Boolean

- Um tipo de dados para representar a satisfação ou não de uma proposição.

- Representa o valor verdade das proposições lógicas

  => **Verdadeiro** ou **Falso**

- O nome é uma homenagem a George Boole que estudou e formalizou as operações com estes tipos de valores.

Definição de funções de verificação

- Agora que temos um novo tipo de dados, podemos utilizá-lo escrevendo expressões de forma tão natural quanto aquela que usamos para escrever expressões aritméticas. Usando essas expressões podemos então construir definições cujo tipo resultante seja booleano.

Verificar se ***a*** é múltiplo de ***b***	`def multiplo(a, b): return (a % b) == 0`
Verificar se ***a*** é divisor de ***b***	`def divisor(a, b): return multiplo(b, a)`
Verificar se 3 números estão em ordem crescente	`def ordc(a, b, c): return (a < b) and (b < c)`

Verificar se um número ***x*** está no intervalo fechado definido por ***a*** e ***b***	`def pert(x,a,b): return (x >= a) and (x <= b)`
	ou
	`def pert(x,a,b): return not((x < a)or(x > b))`

nemo

Definição de funções de verificação

- Podemos usar agora os operadores lógicos para construir expressões compostas:

Verificar se 3 números estão em ordem crescente	`def ordc(a, b, c): return (a < b) and (b < c)`

Verificar se um número ***x*** está no intervalo fechado definido por ***a*** e ***b***	`def pert(x,a,b): return (x >= a) and (x <= b)`
	ou
	`def pert(x,a,b): return not((x < a)or(x > b))`

Verificar se um determinado ponto do espaço cartesiano está no primeiro quadrante	`def pquad(x, y): return (x > 0) and (y > 0)`
Verificar se 3 números ***a***, ***b*** e ***c***, são lados de um triângulo retângulo	`def tret(a, b, c): return ((a**2 + b**2) == c**2) or ((a**2 + c**2) == b**2) or ((b**2 + c**2) == a**2)`

Exemplo

- Desejamos verificar se um determinado ponto do espaço cartesiano está dentro ou fora de um retângulo paralelo aos eixos, conhecidos os pontos: o canto superior esquerdo e o canto inferior direito do retângulo.

Exemplo

- **Etapa 1** [Entendendo o problema]
  - Estamos tratando de qualquer quadrante?
  - De que preciso para definir o retângulo?
  - Um ponto que esteja sobre um dos lados, está dentro ou fora do retângulo?

- Vamos assumir então as seguintes decisões:
  - Ponto e retângulo localizados no primeiro quadrante.
  - O retangulo é definido por canto superior esquerdo e canto inferior direito, representados por pontos.
  - Dos pontos serão informadas as duas coordenadas
  - Os pontos na borda são considerados pertencentes ao retângulo.

Exemplo

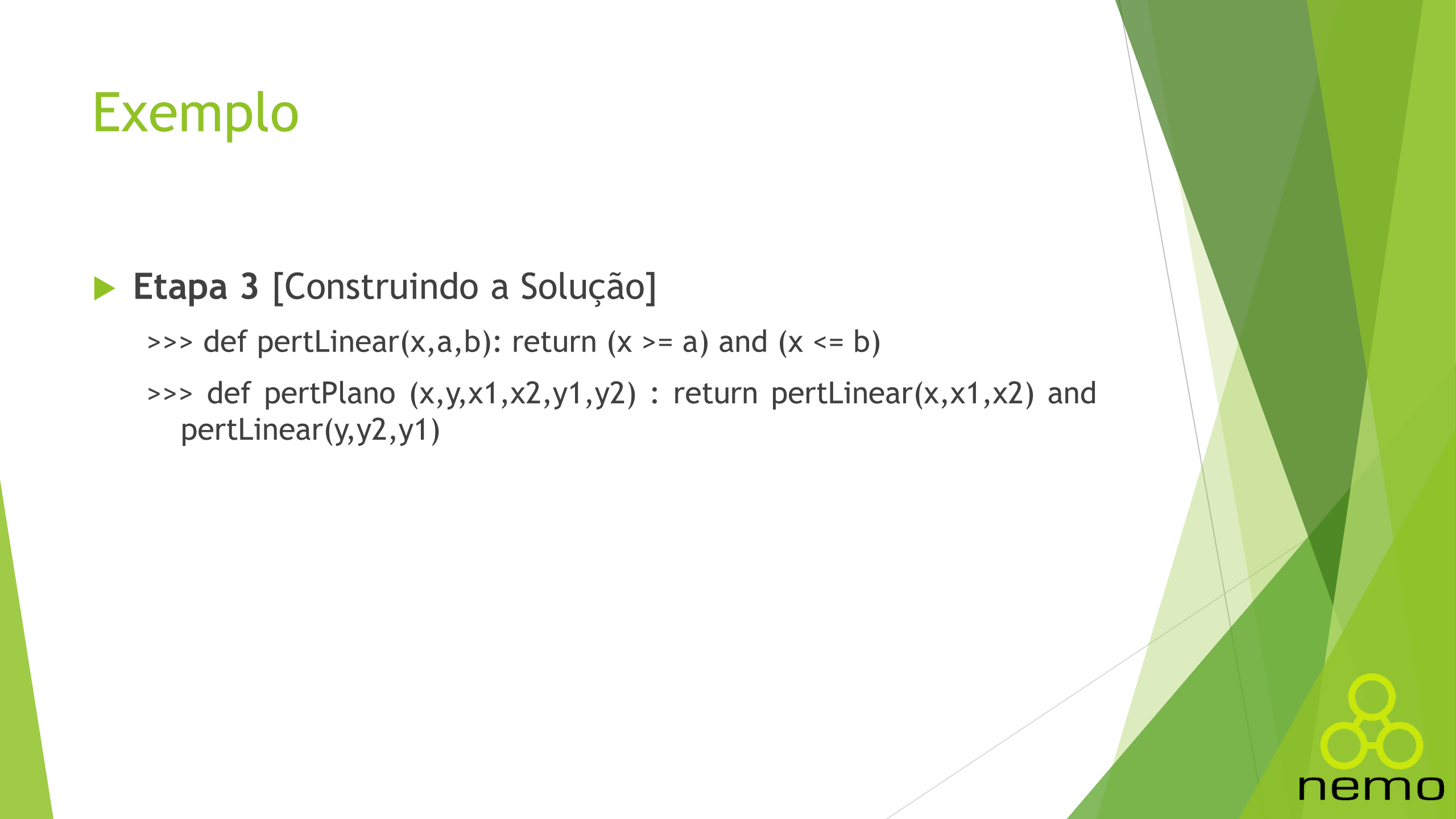
- **Etapa 2** [Planejando a Solução]
  - Conheço algum problema parecido?
  - Posso decompor este problema em problemas mais simples?
  - Sei resolver um problema mais geral de que este é um caso particular?
  - Posso decompor o problema na verificação de dois espaços lineares, um definido pelos lados paralelos ao eixo das ordenadas e outro paralelo ao eixo das abscissas.
  - Como combino as duas soluções?

nemo

Exemplo

- **Etapa 3** [Construindo a Solução]

```
>>> def pertLinear(x,a,b): return (x >= a) and (x <= b)
>>> def pertPlano (x,y,x1,x2,y1,y2) : return pertLinear(x,x1,x2) and
    pertLinear(y,y2,y1)
```



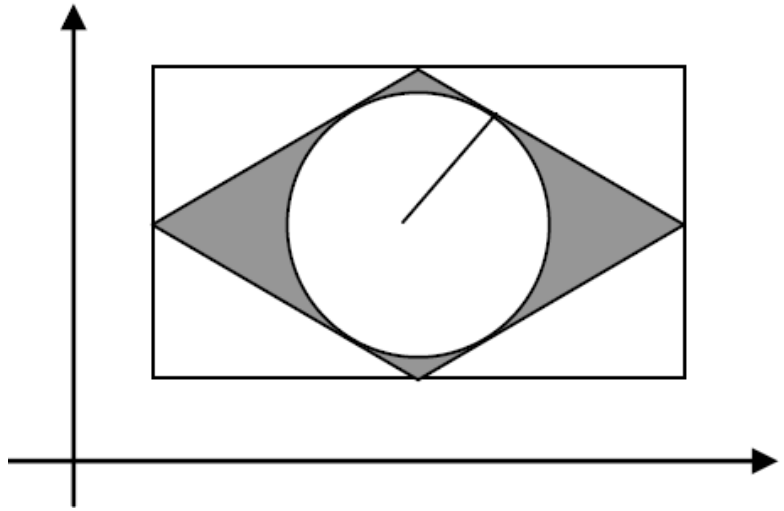
Exemplo

- **Etapa 4** [Analisando a Solução]
  - Existem outras maneiras de resolver o problema?
  - Esta solução se aplica aos outros quadrantes?
  - Esta idéia poderia ser generalizada para outras dimensões?

nemo

Exercício

- Dado um ponto P(x,y) do plano cartesiano, defina funções que descrevam a sua pertinência nas regiões cinzas da figura abaixo:

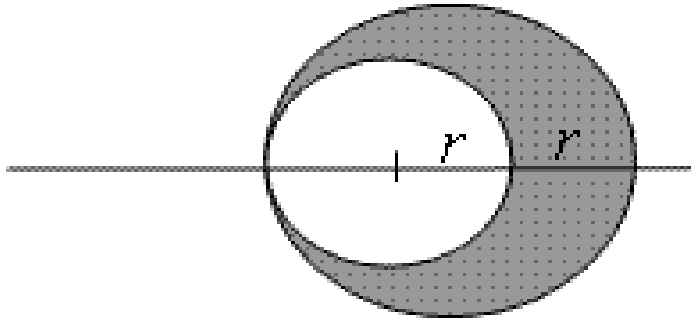


- A região R2 do losango (região cinza), sendo dados os pontos E e D do retângulo e sabendo-se que o círculo é tangente aos lados do losango.

nemo

Exercício

- Descreve funções que calcule:
- A) A raiz da equação ax + b = 0.
- B) A temperatura em graus Farenheit, dada a temperatura em graus Celsius.
  - A proporção é dada por: tc/5 = (tf - 32)/9
- C) A área cinza da figura abaixo, dado o raio *r* do círculo menor



That’s all Folks!