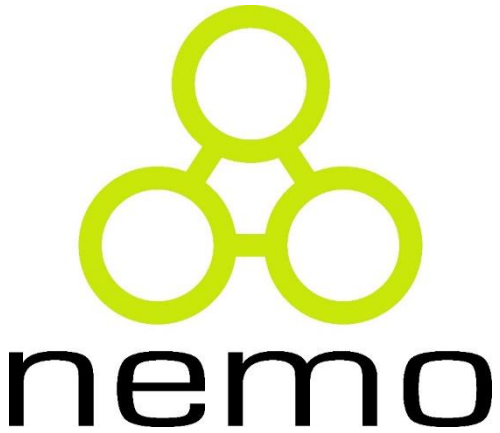


Programação I

Jordana S. Salamon

jssalamon@inf.ufes.br

jordanasalamon@gmail.com

DEPARTAMENTO DE INFORMÁTICA

CENTRO TECNOLÓGICO

UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO

Tipos de Dados

REVISÃO

- 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

Tipo Caracter/String

REVISÃO

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

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

nemo

Expressões lógicas em Python

REVISÃO

- 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

Tipo Boolean

REVISÃO

- 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.

nemo

Definição de funções de verificação

REVISÃO

- 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)`

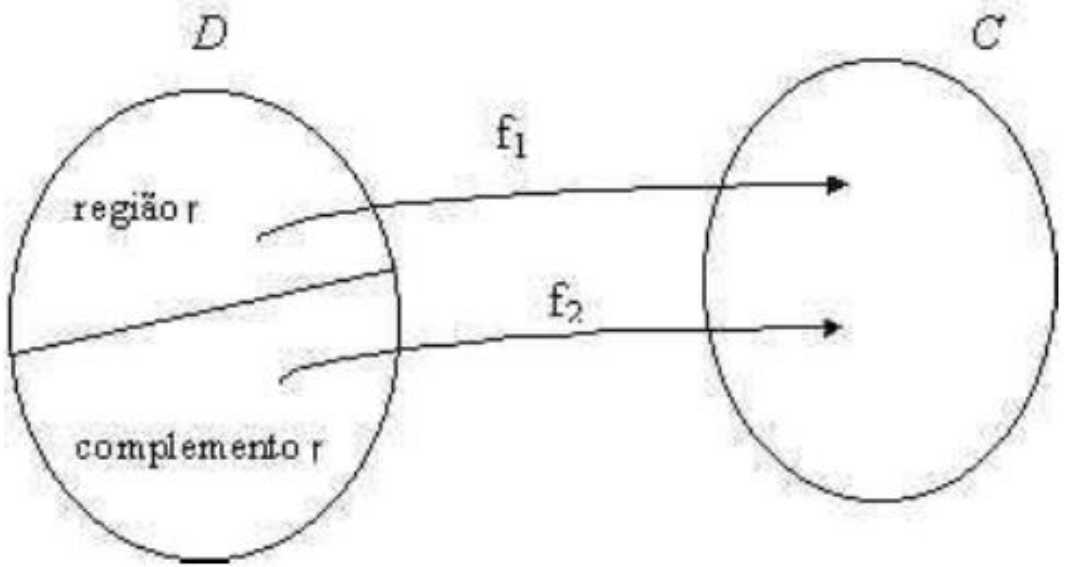
Verificar se um número ***x*** está no intervalo fechado definido por ***a*** e ***b***	`def pert(x,a,b): return (x >= a) and (x <= b)`
	ou
	`def pert(x,a,b): return not((x < a)or(x > b))`

nemo

Definições Condicionais

- Sabemos de nosso conhecimento matemático que algumas funções não são contínuas em um domínio e que, portanto, possuem várias definições.

- Em muitos casos, o domínio D de uma função está dividido em regiões disjuntas que se complementam e, para cada uma dessas regiões, existe uma expressão que define o seu mapeamento no contra-domínio. Podemos representar esta situação pela figura abaixo:



Definições Condicionais

- Exemplo 1 - Considere a função que determina o valor da passagem aérea de um adulto, para um determinado trecho, por exemplo, Vitória-Manaus, considerando a sua idade.

- Pessoas com idade a partir de 60 anos possuem um desconto de 40% do valor. Considere ainda que a passagem para o trecho considerado custe R$ 600,00.

- Temos aqui duas formas de calcular o valor da passagem de uma pessoa, dividindo o domínio em dois subconjuntos.

- O subconjunto dos adultos com menos de 60 anos e o subconjunto dos demais.

- Podemos definir as duas funções a seguir:

Definições Condicionais

- Para usar uma das definições, temos que explicitamente escolher a que se aplica ao nosso caso.

```
>>> def valorPassagem1(): return 600
>>> def valorPassagem2(): return valorPassagem1() * 0.6
```

- **Exemplo 2** - Considere a função que associa com um determinado rendimento o Imposto de Renda a ser pago.

- Até um determinado valor, o contribuinte não paga imposto, e a partir de então o rendimento é dividido em faixas (intervalos), aos quais se aplicam diferentes taxas. Suponha a tabela hipotética abaixo:

Faixa	alíquota	Desconto
inferior ou igual a 10.800	0	0
entre 10.801 e 20.000	10	1000
entre 20.001 e 30.000	20	1500
acima de 30.000	25	1800

Definições Condicionais

- Para descrever as várias definições e os correspondentes subdomínios, poderíamos escrever separadamente cada definição, construindo, portanto várias funções, e deixar que o usuário escolha qual usar.

```
>>> def imposto1(s): return 0
>>> def imposto2(s): return s * 0.1 - 1000
>>> def imposto3(s): return s * 0.2 - 1500
>>> def imposto4(s): return s * 0.25 - 1800
```

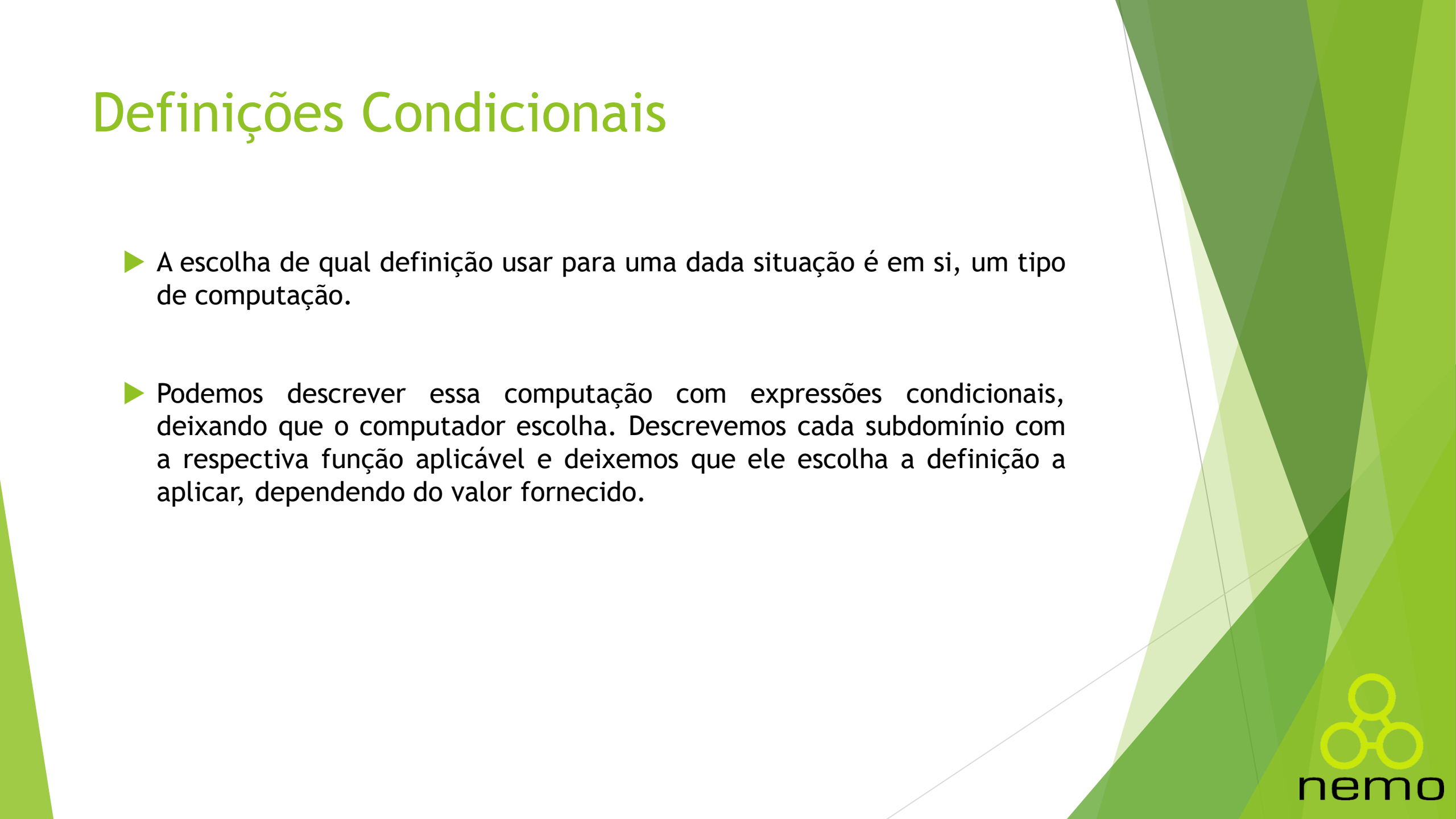
- Agora, para usá-las, o usuário pega o seu salário, olha a tabela e seleciona qual função aplicar.

nemo

Definições Condicionais

- A escolha de qual definição usar para uma dada situação é em si, um tipo de computação.

- Podemos descrever essa computação com expressões condicionais, deixando que o computador escolha. Descrevemos cada subdomínio com a respectiva função aplicável e deixemos que ele escolha a definição a aplicar, dependendo do valor fornecido.



Definições Condicionais

- Vamos estudar inicialmente a construção mais simples de definição condicional, construída com a estrutura **if-else** segundo a seguinte sintaxe:

```
if <expressão lógica>: <expressão 1>
else: <expressão 2>
```

- Onde:

<expressão lógica>	Uma expressão descrevendo uma condição a ser satisfeita, envolvendo operadores relacionais e operadores lógicos.
<expressão1> e <expressão2>	1. Expressões descrevendo um valor a ser produzido como resposta à entrada fornecida e, como a expressão total deve ser de um único tipo, as duas expressões devem ser do mesmo tipo. 2. Cada uma destas expressões pode ser inclusive outra condicional, dentro da qual pode haver outras e assim sucessivamente. 3. Quando a <expressão lógica> é avaliada para ***True*** o valor resultante será o que for obtido pela avaliação da <expressão 1> caso contrário será o obtido pela avaliação da <expressão 2>

nemo

Definições Condicionais

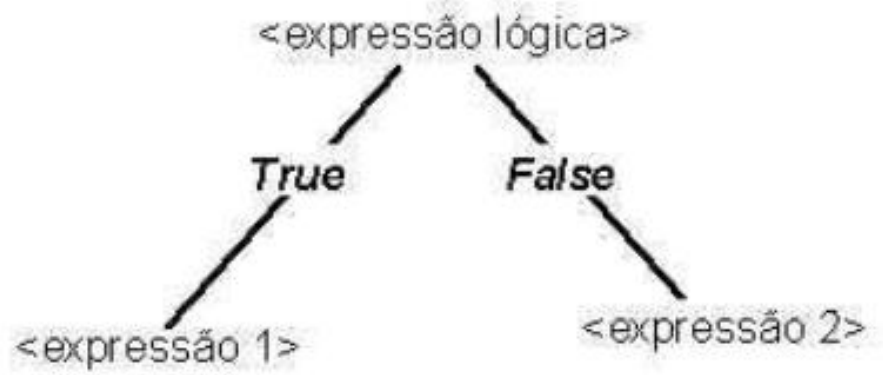
- Para a função que calcula o valor da passagem aérea podemos então construir a seguinte definição:

```
>>> def valorPassagem(x):
        if x < 60: return 600
        else: return 360
```

nemo

Definições Condicionais

- Podemos representar as expressões condicionais através de uma notação gráfica denominada de árvore de decisão. É importante considerar que este tipo de representação é uma ferramenta importantíssima para estruturarmos a solução de problemas que requerem expressões condicionais.



nemo

Definições Condicionais

- Exemplo 3 – Definir a função que determina o valor absoluto de um número. Sabemos que esta função se define em dois subdomínios:

subdomínio	expressão
x < 0	- x
x >= 0	x

- Como só temos duas possibilidades, podemos codificar da seguinte maneira:

```
>>> def absoluto(x):
        if x < 0: return -x
        else: return x
```

Definições Condicionais

- Contudo, alguns problemas apresentam uma definição condicional mais complexa, de forma que o problema se divide em mais de dois subdomínios.

- Nestes casos, a estrutura **if-elif-else**, complementar à recém-apresenta, deve ser usada segundo a seguinte sintaxe:

```
if <expressão lógica 1>: <expressão 1>
elif <expressão lógica 2>: <expressão 2>
...
elif <expressão lógica n-1>: <expressão n-1>
else: <expressão n>
```

nemo

Definições Condicionais

- Este é o caso do exemplo 2 que define a função para cálculo do Imposto de Renda.

- O domínio neste caso deve ser quebrado em quatro subdomínios e para cada um deles construiremos uma expressão.

Domínio	Expressão
$s \leq 10800$	0
$10800 < s \leq 20000$	s * 0.1 - 1000
$20000 < s \leq 30000$	s * 0.2 - 1500
$s > 30000$	s * 0.25 - 1800

nemo

Definições Condicionais

- Para a codificação, precisaremos quebrar sucessivamente o domínio da função em intervalos.
- A codificação final pode ser:

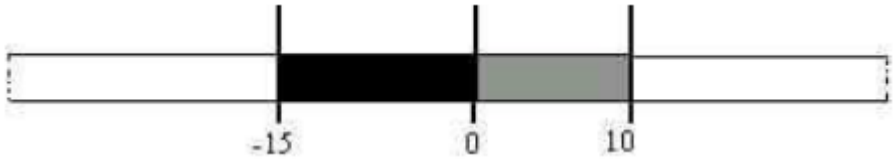
```
>>> def imposto(s):
        if s <= 10800: return 0
        elif s <= 20000: return s * 0.1 - 1000
        elif s <= 30000: return s * 0.2 - 1500
        else: return s * 0.25 - 1800
```

nemo

Definições Condicionais

- Exercício: Defina a função que resolve o problema proposto

1) Considere um mapeamento de valores numéricos onde o domínio se divide
em 4 regiões, cada uma das quais possui diferentes formas de mapeamento. As regiões são apresentadas na figura abaixo, numeradas da esquerda para direita. Observe ainda que as extremidades são abertas. Considere ainda a seguinte tabela de associações:



região	mapeamento desejado
região 1	o dobro de x
região 2	o sucessor de x
região 3	o quadrado de x
região 4	o simétrico do quadrado de x

2) Dados três números inteiros distintos, determinar o maior deles.

nemo

That’s all Folks!