Fundamentos da Programação

Aula 10

Funções

Visualização e execução de programas. Depuração. Exemplos.

Alberto Abad, IST, 2024-25

Visualização e execução de programas

- **http://pythontutor.com/visualize.html#mode=edit**
- IDEs como o **Visual Studio Code**, PyCharm e WingIDE

A treinar mais!!!!

Exemplo 4, Máximo divisor comum (Algoritmo de Euclides)

1. O máximo divisor comum entre um número e zero é o próprio número:

`mdc(m,0) = m` 2. Quando dividimos um número *m* por *n*, o máximo divisor comum entre o resto da divisão e o divisor é o mesmo que o máximo divisor comum entre o dividendo e o divisor: `mdc(m, n) = mdc(n, m%n)`

- Exemplo algorimo para mdc(24, 16):

m	n	m % n
24	16	8
16	8	0
8	0	8

Exemplo 4, Máximo divisor comum (Algoritmo de Euclides)

In [22]:
```
# Máximo divisor comum (mdc)
# Euclidian algorithm

def mdc(m, n):
    pass

x = eval(input("Da-me valor x:"))
y = eval(input("Da-me valor y:"))
print(mdc(x, y))
```

```
Da-me valor x:16
Da-me valor y:24
8
```

Exemplo 5, Raiz quadrada (Algoritmo da Babilónia)

- Em cada iteração, partindo do valor aproximado, p_i, para a raiz quadrada de x, podemos calcular uma aproximação ao melhor, p_{i+1}, através da seguinte fórmula:

$$p_{i+1} = \frac{p_i + \frac{x}{p_i}}{2}.$$

- Exemplo algoritmo para $\sqrt{2}$

Número da tentativa	Aproximação para $\sqrt{2}$	Nova aproximação
0	1	$\frac{1+\frac{2}{1}}{2} = 1.5$
1	1.5	$\frac{1.5+\frac{2}{1.5}}{2} = 1.4167$
2	1.4167	$\frac{1.4167+\frac{2}{1.4167}}{2} = 1.4142$
3	1.4142	...

Exemplo 5, Raiz quadrada (Algoritmo da Babilónia)

```python
def calcula_raiz(x, palpite):
    while not bom_palpite(x, palpite):
        palpite = novo_palpite(x, palpite)
    return palpite
```

```python
def raiz(x):
    if x < 0:
        raise ValueError("raiz definida só para números positivos")
    return calcula_raiz(x, 1)
```

- **Exercício**: Definir as funções *bom_palpite* e *novo_palpite*

```python
In [26]: def calcula_raiz(x, palpite):
             while not bom_palpite(x, palpite):
        palpite = novo_palpite(x, palpite)
    return palpite
def raiz(x):
    if x < 0:
        raise ValueError("raiz definida só para números positivos")
    return calcula_raiz(x, 1)

def bom_palpite(x, palpite):
    pass

def novo_palpite(x, palpite):
    pass

import math

x = 2
print("Aprox ",raiz(x))
print("Exacto",math.sqrt(x))
```

```
Aprox  1.414213562373095
Exacto 1.4142135623730951
```

Exemplo 6, Séries de Taylor

- Definição:

$$f(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!}(x-a)^n = f(a) + \frac{f'(a)}{1!}(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \frac{f^{(3)}(a)}{3!}(x-a)^3 + \cdots$$

- Exemplos dalgumas aproximações:

$$\begin{aligned} e^x &= \sum_{n=0}^{\infty} \frac{x^n}{n!} = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots \\ \sin(x) &= \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n+1)!} x^{2n+1} \quad = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \cdots \\ \cos(x) &= \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n)!} x^{2n} \quad = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \cdots \end{aligned}$$

Exemplo 6, Séries de Taylor

```
def proximo_termo(x, n):
    pass #completar (diferente dependendo da função a aproximar)
```

```
def funcao_aproximada(x, delta):
    n = 0
    termo = proximo_termo(x, n)
    resultado = termo
    while abs(termo) > delta:
        n = n + 1
        termo = proximo_termo(x, n)
        resultado = resultado + termo
    return resultado
```

- **Exercício:** Definir a série de Taylor para as fuņçoes *e(x)*, *sin(x)* e *cos(x)*
- **Exercício:** Alterar para que o cômputo de termo seja função do anterior termo, `termo = proximo_termo(x, n, termo)`

Exemplo 6, Séries de Taylor: Exponencial

In [13]:
```
def proximo_termo(x, n):
    pass


def exp_aproximada(x, delta):
    n = 0
    termo = proximo_termo(x, n)
    resultado = termo

    while abs(termo) > delta:
        n = n + 1
        termo = proximo_termo(x, n)
        resultado = resultado + termo

    return resultado

import math

print("Aprox ",exp_aproximada(0,0.01))
print("Exacto",math.exp(0))
```

```
Aprox  1.0
Exacto 1.0
```

Exemplo 6, Séries de Taylor: Seno

In [16]:
```
def proximo_termo(x, n):
    pass

def sin_aproximada(x, delta):

    n = 0
    termo = proximo_termo(x, n)
    resultado = termo

    while abs(termo) > delta:
        n = n + 1
        termo = proximo_termo(x, n)
        resultado = resultado + termo

    return resultado

import math

print("Aprox",sin_aproximada(math.pi/2,0.0001))
print("Exacto",math.sin(math.pi/2))
```

```
Aprox 0.999999943741051
Exacto 1.0
```

Exemplo 6, Séries de Taylor: Cosseno

In [8]:
```
def proximo_termo(x, n):
    pass

def cos_aproximada(x, delta):
    n = 0
    termo = proximo_termo(x, n)
    resultado = termo

    while abs(termo) > delta:
        n = n + 1
        termo = proximo_termo(x, n)
        resultado = resultado + termo

    return resultado

import math

print("Aprox",cos_aproximada(math.pi/6,0.0001))
print("Exacto",math.cos(math.pi/6))
```

```
Aprox 0.8660252641005711
Exacto 0.8660254037844387
```

Funções - Tarefas para a próxima semana

- Trabalhar matéria apresentada até hoje --> Fazer todos os programas!
- Ler capítulo 4 do livro da UC: Tuplos, ciclos contados e cadeias de carateres
- Na próxima **aula laboratorial (L05)**:
    - Ficha F2 - Elementos básicos de programação
    - L05: Funções, verificação de argumentos, exepções