Aula 01: 02/AGO/2018

Resumo de MAC0110/0115/2166

Até agora, mais ou menos, conversamos sobre:

- classes nativas: `int`, `float`, `bool`, `list`, `str`, `Nonetype`
- funções
- listas, listas de listas, vetores, matrizes
- strings
- percorrer listas, strings, vetores, matrizes
- apelidos versus clones
- **sobretudo**: raciocínio aplicado a resoluções de problemas computacionais... blá-blá-blá

Hoje

Começamos a trilhar um caminho para uma *introdução rudimentar à programação orientada a objetos* e tratar de tópicos como:

- objetos: referências
- classes nativas *versus* classes definidas pelo usuário
- atributos de estado e métodos
- método especial/mágico construtor `__init__()`
- método especial/mágico `__str__()`
- objetos são mutáveis
- sobrecarga de operadores e os métodos especiais/mágicos: `__add__()`, `__mul__()`, ...
- doce sintático (= *syntactic suger*)

Problema (versão aproximada)

Dado um inteiro positivo `n`, calcular o valor de H_n, o número *harmônico de ordem* `n` de duas maneiras:

```
1 +    1/2     +   1/3     + 1/4 + ... + 1/n
1/n +  1/(n-1) +   1/(n-2) + ... + 1/2 + 1
```

Observação 1: no semestre passado discutimos o cálculo do números harmônicos de ordem `n` da direita para a esquerda e da esquerda para a direita.

Na ocasião o objetivo era falarmos sobre `floats` (com sua precisão limitada) versus ‘`ints` (potencialmente ilimitados).

Hoje, essa discussão nos levará a implimentação da classe `Fracao`.

Observação 2: pode ser usado para lembrar funções e os cálculos aproximados envolvendo `floats`.

Solução

Solução usando `float`, ou seja, aproximada.

```
def main():
    n = int(input("Digite n: "))

    hn1 = harmonico_ED(n)
    print("1 + ... + 1/%d + 1/%d = " %(n-1,n), hn1)

    hn2 = harmonico_DE(n)
    print("1/%d + 1/%d + ... + 1 = " %(n,n-1), hn2)

#-----------------------------------------------------
def harmonico_ED(n):
    '''(int) -> float

    Recebe um inteiro n e retorna o número harmônico
    de ordem n que foi calculado adicionando-se os termos
    da esquerda para a direita.
    '''
    soma = 0
    i = 1
    while i < n+1:
        soma += 1/i
        i += 1
    return soma


#-----------------------------------------------------
def harmonico_DE(n):
    '''(int) -> float

    Recebe um inteiro n e retorna o número harmônico
    de ordem n que foi calculado adicionando-se os termos
    da direita para a esquerda.
    '''
    soma = 0
    i = n
    while i > 0:
        soma += 1/i
        i -= 1
    return soma


#-----------------------------------------------------
# início da execução do programa
main()

#-----------------------------------------------------
def main():
    n = int(input("Digite n: "))

    soma = 0
```

```
    i = 1
    while i < n+1:
        soma += 1/i
    i += 1
    print("1   +   1/2   + ... + 1/n = ", soma)

    soma = 0
    i = n
    while i > 0:
        soma += 1/i
    i -= 1
    print("1/n + 1/(n-1) + ... +  1  = ", soma)
```

Exemplos de execução

```
Digite n: 1000000
1   +   1/2   + ... + 1/n =  14.392726722864781
1/n + 1/(n-1) + ... +  1  =  14.392726722865754

Digite n: 6
1   +   1/2   + ... + 1/n =  2.4499999999999997
1/n + 1/(n-1) + ... +  1  =  2.45
```

Representação de frações

Podemos representar uma fração de maneira exata através do seu *numerador* e o seu *denominador*:

```
1/3 em vez de 0.3333333
2/4 em vez de 0.5
```

Problema (versão exata)

Dado um inteiro positivo `n`, calcular o valor de H_n, o número *harmônico de ordem* `n`:

```
1 +    1/2    +   1/3   + 1/4 + ... + 1/n
```

de **maneira exata** usando frações.

Exemplos

```
Digite n: 6
1   +   1/2   + ... + 1/n =  49/20

Digite n: 7
1   +   1/2   + ... + 1/n =  363/140
>>> 363/140
2.592857142857143
```

```
Digite n: 7
1   +   1/2   + ... + 1/n =  2.5928571428571425
1/n + 1/(n-1) + ... +  1  =  2.5928571428571425

Digite n: 15
1   +   1/2   + ... + 1/n =  3.3182289932289937
1/n + 1/(n-1) + ... +  1  =  3.318228993228993

Digite n: 15
1   +   1/2   + ... + 1/n =  1195757/360360
>>> 1195757/360360
3.3182289932289932
```

Solução

Basta calcular da esquerda para a direita ou da direita para a esquerda.

```
def main():
    n = int(input("Digite n: "))

    n1, d1 = harmonico_FracED(n)
    print("1 + ... + 1/%d = %d/%d = %f" %(n,n1,d1,n1/d1))

    n2, d2 = harmonico_FracDE(n)
    print("1/%d + ... + 1 = %d/%d = %f" %(n,n2,d2,n2/d2))

#---------------------------------------------------------------
def harmonico_FracED(n):
    ''' (int) -> int, int

    Recebe um numero inteiro positivo e retorna o numero harmonico
    de ordem n representado como uma racional.
    O numero harmonico e calculado somando os termos
    da esquerda para a direita.
    '''
    num, den = 0, 1 # numerador e denominador
    for i in range (1, n+1):
        num, den = soma_fracoes(num,den,1,i)
    return num, den

#---------------------------------------------------------------
def harmonico_FracDE(n):
    ''' (int) -> int, int

    Recebe um numero inteiro positivo e retorna o numero harmonico
    de ordem n representado como uma fração.
    O número harmônico e calculado somando os termos da direita
    para a esquerda.
    '''
```

```
    num, den = 0, 1 # numerador e denominador
    for i in range (n, 0, -1):
        num, den = soma_fracoes(num,den,1,i)
    return num, den

#-------------------------------------------------------------
def soma_fracoes(n1, d1, n2, d2):
    ''' (int, int, int, int) -> int, int

    Recebe quatro numeros inteiros n1, d1, n2 e d2 representando
    duas fracoes n1/d1 e n2/d2 e retorna um par (num,den)
    que representa a soma desses números.

    Pre-condicao: a função supõe que os quadro números dados nao
    sao nulos.
    '''
    den = d1 * d2
    num = n1*d2 + n2*d1
    return simplifique(num, den)


#-----------------------------------------
def simplifique(n, d):
    '''(int, int) -> int, int

    Recebe uma racional n/d e retorna a correspondente
    racional irredutível.
    '''
    comum = mdc(n,d)
    n //= comum  # precisa ser //
    d //= comum  # precisa ser //
    if d < 0:
        d = -d
        n = -n
    return n, d

#---------------------------------------
def mdc(m,n):
    """ (int, int) -> int
    Recebe dois inteiros m e n e retorna o
    mdc de m e n.
    Pre-condiçao: a funçao supoe que n != 0
    """
    if m < 0: m = -m
    if n < 0: n = -n
    if n == 0: return m
    r = m%n
    while r != 0:
        m = n
        n = r
```

```
        r = m % n
    return n


#------------------------------------------------
# início da execução do programa
main()
```

Nota: a diferença de consumo de tempo é gritante se usarmos

```
#----------------------------------------
def mdc(m,n):
    """ (int, int) -> int
    Recebe dois inteiros m e n e retorna o
    mdc de m e n.
    Pre-condiçao: a funçao supoe que min(abs(n),abs(m)) != 0
    """
    if m < 0: m = -m # m = abs(m)
    if n < 0: n = -n # n = abs(n)
    d = min(n, m)
    if d == 0: return m
    while m % d != 0 or n % d != 0:
        d -= 1
    return d
```