Recursão

- Um subprograma (subrotina) é recursivo se ele é definido em termos de si mesmo.
  - Ele possui dentro de seu corpo de comandos uma ou mais chamadas a si mesmo.
  - Obviamente que com parâmetros diferentes, senão ele ficaria em loop infinito.
- No caso da recursão em subprogramas:
  - A condição básica será a condição de parada das chamadas recursivas – o resultado é conhecido.
  - A redução do problema será feita escrevendo o cálculo atual em função do resultado do valor anterior, desde que não seja a condição básica.

Observações importantes

- Já foi provado que tudo que se faz usando recursão também se faz sem recursão, usando comandos de repetição.
  - Nos casos de recursão linear (em um único caminho), as duas soluções são “equivalentes” em termos de eficiência ou complexidade.
- Quando a recursão se dá em mais de um caminho, os subprogramas recursivos tendem a ser muito menores e mais intuitivos do que os não recursivos.
  - Caso por exemplo dos algoritmos para implementação de árvores.

```
#Cálculo do fatorial – subprograma usando repetição:
def fatorial (num) :
    f = 1
    for i in range (2, num + 1) :
        f = f * i
    return f
```

```
fat = fatorial (5)

#Cálculo do fatorial – subprograma recursivo:
def fatorial (num) :
    if num < 2 :
        f = 1
    else :
        f = num * fatorial (num – 1)
return f

fat = fatorial (5)
```

- Problemas de cálculo de séries matemáticas em geral também podem ser resolvidos facilmente usando subrotinas recursivas
  - O número de termos pode ser usado como base para controlar a recursão.
  - Parâmetros auxiliares podem ser utilizados para levar resultados ou valores para as chamadas recursivas.
    - Neste caso, pode ser interessante usar um outro nome para esconder estes parâmetros dos usuários.

```
#Exemplo: S = 1 + 3/2 + 5/3 + 7/4 + ... (n termos)
def serieR (n, nu = 1, de = 1.0) :
    if n <= 1 :
        res = nu / de
    else :
        res = nu / de + serieR (n – 1, nu + 2, de + 1)
    return res

#Solução alternativa:
def serieR (n, nu = 1, de = 1.0) :
    res = nu / de
    if n > 1 :
        res = res + serieR (n – 1, nu + 2, de + 1)
    return res
```

```
#Se quiser esconder os parâmetros adicionais…
def serie (n) :
    return serieR (n, 1, 1.0)
```

Exercícios

1. Fazer subrotinas recursivas para calcular o valor das seguintes séries – com n termos, onde n deve ser um parâmetro recebido na chamada:
   – S2 = 37*
   38/1 - *36**37/2 + 35**36/3* – *34**35/4 + 33∗34/5 - 32∗33/6 + 31∗32/7 - 30∗31/8 ...
   – S3 = 2/500 – 5/490 + 2/480 – 5/470 + ...
   • OBS: Mostrar como será a chamada externa do método, por exemplo, no programa principal.
2. Fazer um programa para:
   – Ler números inteiros positivos digitados pelo usuário.
   • A leitura pára quando um número negativo for digitado.
   – Imprimir, para cada um deles, o termo equivalente na seqüência de Fibonacci.
   – Os números da seqüência de Fibonacci são:
   • 0, 1, 1, 2, 3, 5, ... Note que a partir do terceiro termo, cada termo é a soma dos 2 termos anteriores.
   – Usar uma subrotina que recebe o número do termo como parâmetro e retorna o seu valor.
   • implementá-la nas versões recursiva e não recursiva.

```
# 1. Fazer subrotinas recursivas para calcular o valor das segui
# onde n deve ser um parâmetro recebido na chamada:
# – S2 = 37*38/1 - 36*37/2 + 35*36/3 – 34*35/4 + ...
# – S3 = 2/500 – 5/490 + 2/480 – 5/470 + ...
# • OBS: Mostrar como será a chamada externa do métodom, por exe

def subrotina1(n, numerador = 37, denominador = 1):
    s2 = (numerador * (numerador + 1)) / denominador
```

```
    if n > 1:
        s2 -= subrotina1(n - 1, numerador - 1, denominador + 1)
    return s2

def subrotina2(n, numerador = 2, denominador = 500):
    s3 = numerador/denominador
    if n%2 ==0:
        numerador = -5
    else:
        numerador = 2
    if n > 1:
        s3 += subrotina2(n - 1, numerador, denominador - 10)
    return s3

n = int(input("Digite o número de termos da sequência:"))
res = subrotina1(n)
res2 = subrotina2(n)
print(f"As sequências com {n} termos são {res} e {res2}")

# Fazer um programa para:
# – Ler números inteiros positivos digitados pelo usuário.
# • A leitura pára quando um número negativo for digitado.
# – Imprimir, para cada um deles, o termo equivalente na seqüênc
# – Os números da seqüência de Fibonacci são:
# • 0, 1, 1, 2, 3, 5, ... Note que a partir do terceiro termo, c
# – Usar uma subrotina que recebe o número do termo como parâmet
# • implementá-la nas versões recursiva e não recursiva.

def fibonacci(n):
    if n == 1 or n == 2:
        f = 1
    else:
        f = fibonacci(n-1) + fibonacci(n -2)
    return f

n = int(input("Digite um número:"))
```

```
while n> 0:
    res = fibonacci(n)
    print(f"\nO termo {n} de Fibonacci é {res}.")
    n = int(input("Digite um número:"))
```