14 Reunião 14: 13/OUT/2020

14.1 Reuniões passadas

Mais funções:

- `binomial()`
- linha do triângulo de pascal: `binomial()` e `fatorial()`
- frações irredutíveis: função `mdc()` e
- EP06: função `primo()` e `goldbach()` e ...
- variáveis locais
- parâmetros
- Em Python **toda função** retorna algum valor

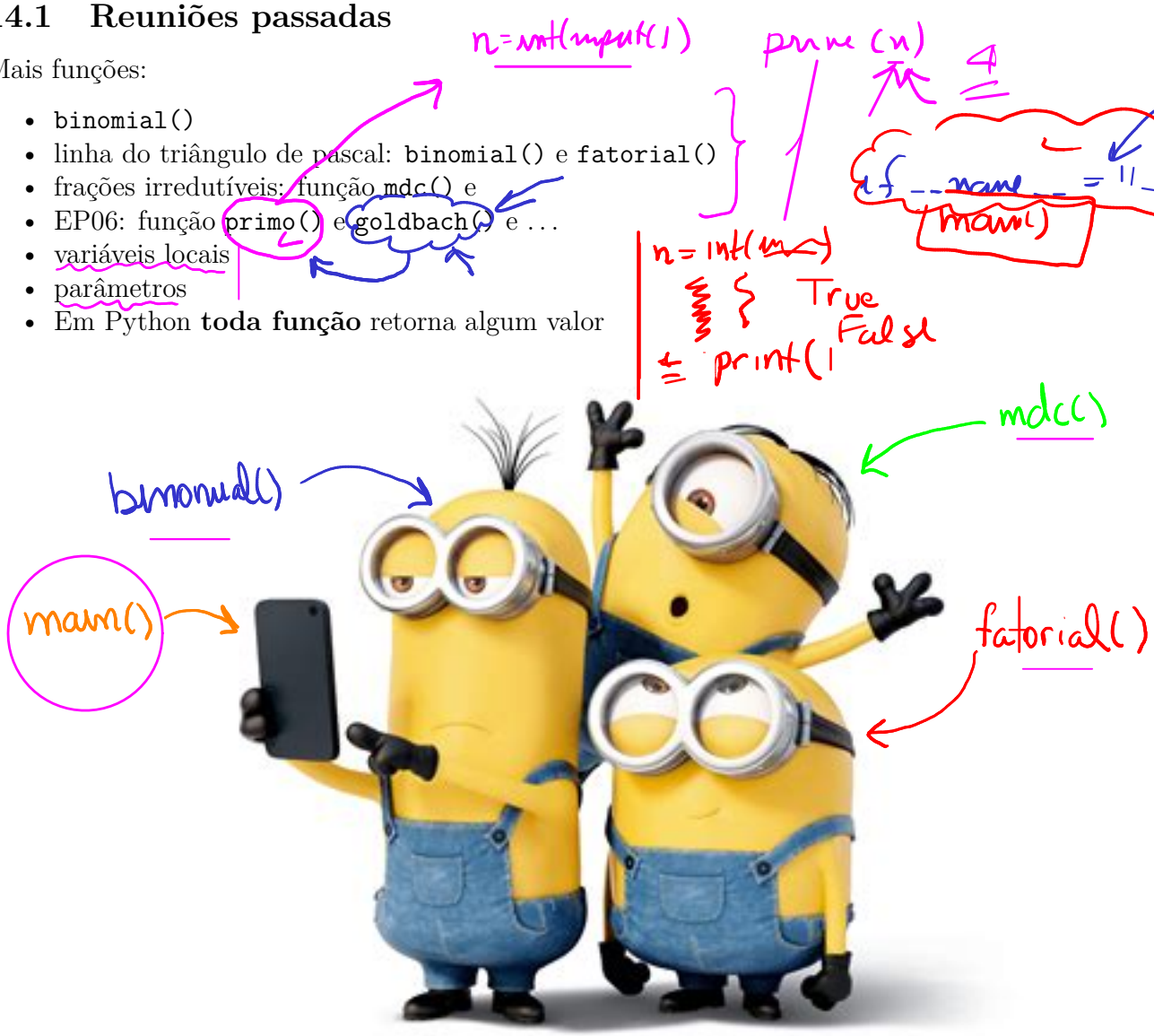
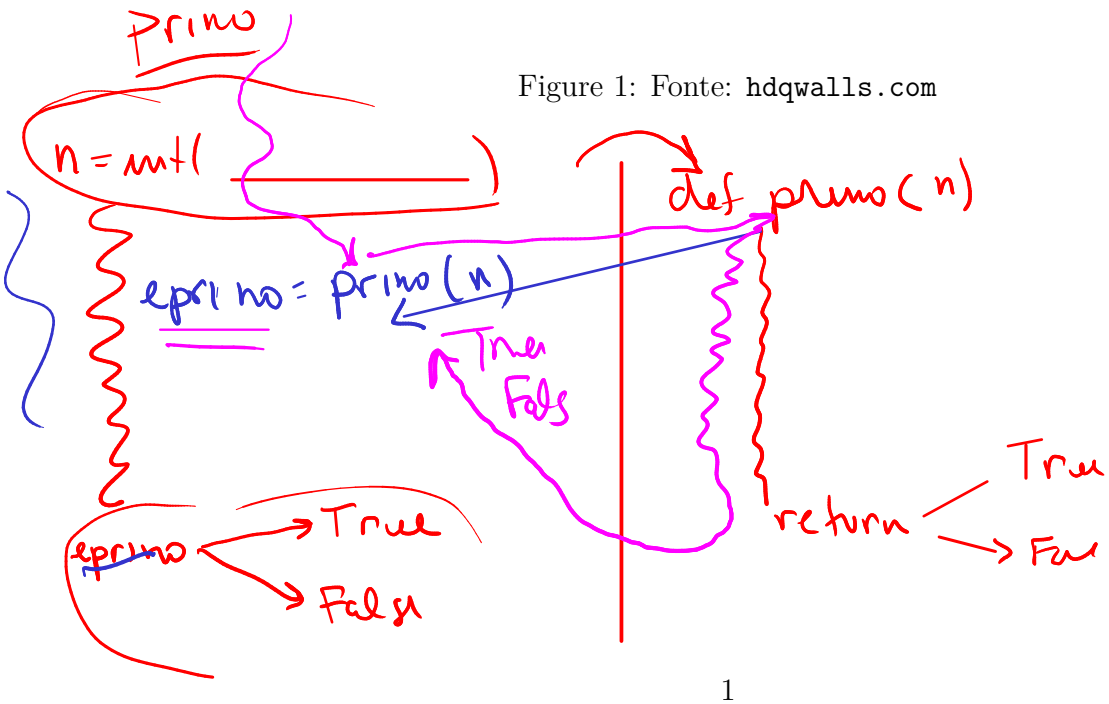


Figure 1: Fonte: `hdqwalls.com`



14.2 Tipos e constantes

```
float    : 3.14, 2.79
int      : 123, -26, 6787
bool     : True False
str      : "Bom dia!"  'Acorda!'
NoneType : None  # ESSE CARA É NOVO!
```

14.3 Leitura

```
f      = float(input("mensagem")) # float
f      = float(input(PROMPT)) # float

n      = int(input("mensagem"))   # int
n      = int(input(PROMPT))   # int

texto = input("mensagem")         # str
texto = input(PROMPT)         # str
```

14.4 Impressão:

Podemos usar os *marcadores*:

- `%f` para `float` ou f-string `f"{x}`, `x` é do tipo `float`
- `%d` para `int` ou f-string `f"{i}`, `i` é do tipo `int`
- `%s` para `str` ou f-string `f"{s}`, `s` é do tipo `str`

Exemplos

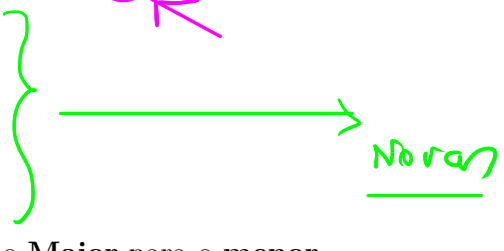
```
print(f"A média das {n} provas é {média}")

print("A média das", n, " provas é", média)

print("A média das %d provas é %f" %(n,média))

print(f"{x}")

print("%f"%(x))

print("x=%f e n=%d\n" %(x,n))
```

14.5 Exercício: números harmônicos

Calcular H_n, o **número harmônico de ordem** `n`, de duas maneiras diferentes.

$$H_n = 1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \cdots + \frac{1}{n}.$$

$$H_n = \frac{1}{n} + \frac{1}{n-1} + \cdots + \frac{1}{2} + 1.$$



Uma maneira adicionando-se os termos do **Maior** para o **menor** e outra do **menor** para o **Maior**.

14.5.1 Exemplos

```
Digite n: 10
1 + ... + 1/9 + 1/10 =  2.9289682539682538
1/10 + 1/9 + ... + 1 =  2.9289682539682538

Digite n: 20
1 + ... + 1/19 + 1/20 =  3.597739657143682
1/20 + 1/19 + ... + 1 =  3.597739657143682

Digite n: 30
1 + ... + 1/29 + 1/30 =  3.9949871309203906
1/30 + 1/29 + ... + 1 =  3.9949871309203915
```

14.5.2 Rascunhos

(a) Escreva uma função `harmonico_Mm()` que **recebe** `n` e retorna o valor de H_n adicionando-se os termos do **M**aior para o **m**enor.

(b) Escreva uma função `harmonico_mM()` que **recebe** `n` e retorna o valor de H_n adicionando-se os termos do **m**enor para o **M**aior.

(c) Escreva uma função `main()` que lê um número inteiro positivo `n` e imprime os valores de H_n obtidos adcionando os termos do Maior para o $m$enor e do $m$enor para o Maior.

14.5.4 Solução

(a) `harmonico_Mm()` **recebe** `n` e retorna o valor de

$$H_n = 1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \cdots + \frac{1}{n}.$$

```
def harmonico_Mm(n):
    '''(int) -> float

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

(b) `harmonico_mM()` **recebe** `n` e retorna o valor de

$$H_n = \frac{1}{n} + \frac{1}{n-1} + \cdots + \frac{1}{2} + 1.$$

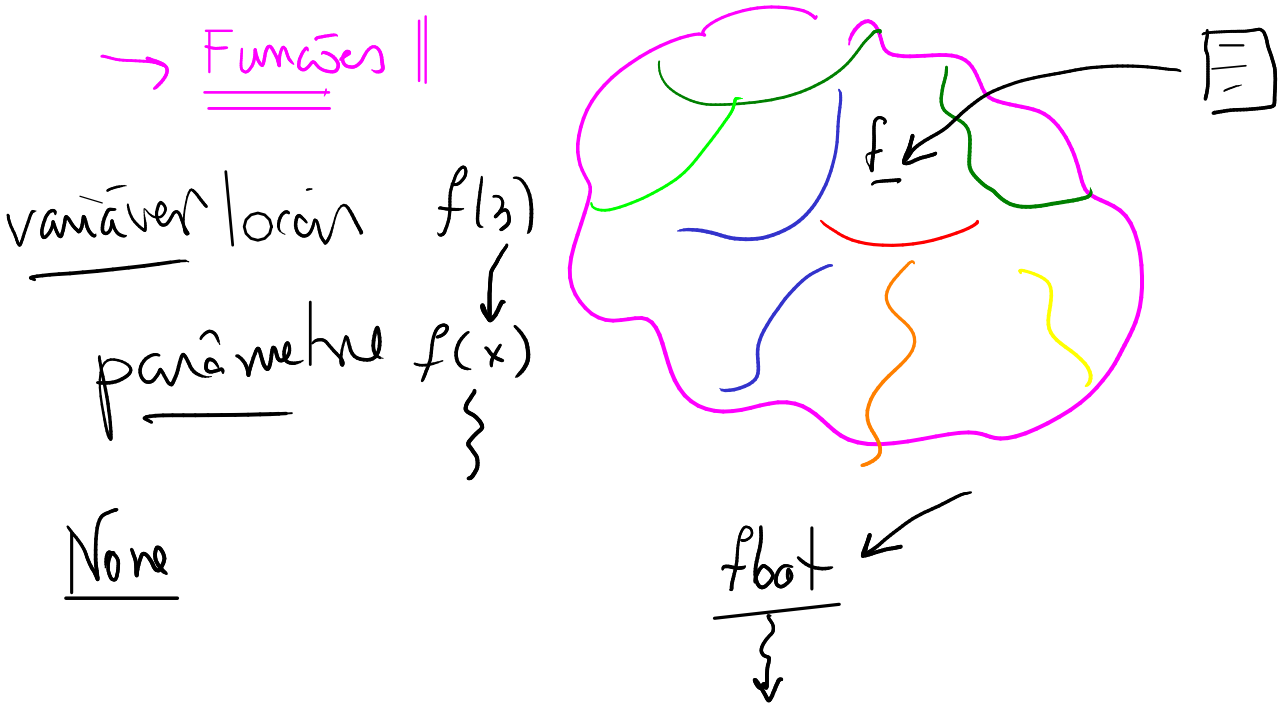
```
def harmonico_mM(n):
    '''(int) -> float

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

(c) `main()` lê um número inteiro positivo `n` e imprime os valores de H_n obtidos adicionando os termos do **M**aior para o **m**enor do **m**enor para o **M**aior.

```
def main():
    '''(None) -> None

    Lê um inteiro n e imprime o valor de

    1 + 1/2 + 1/3 + ... + 1/n

    Calculado do maior para o menor e
    vice-versa
    '''
    n = int(input("Digite n: "))

    h_Mm = harmonico_Mm(n)
    print(f"1 + ... + 1/{n-1} + 1/{n}= {h_Mm}")

    h_mM = harmonico_mM(n)
    print(f"1/{n} + 1/{n-1} + ... + 1={h_mM}")
```

14.5.5 Mais 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
```



14.6 `float` versus `int` em Python

Em Python um valor `int` (nativo) é potencialmente ilimitado.

Análise numérica estuda a instabilidade numérica de algoritmos (= aproximações calculadas por programas).

Segue blá-blá-blá do prof.

Um valor `float` é uma *aproximação* de uma fração (= número racional).

Utilizam representação em pontos flutuante.

Em Python um `float` tem aprox. 16 dígitos decimais:

```
8 bits = 1 byte

Tamanhos de double:      (8 bytes)
menor valor positivo:    2.2250738585072014e-308
maior valor:             1.7976931348623157e+308
1 bit para o sinal
11 bits para o expoente
52 bits para o número
aprox. 15 dígitos de precisão
```