MAC 110 — Introdução à Ciência da Computação

Aula 19

Nelson Lago

BMAC – 2024

Prova

Prova — questão 1

Simule a execução da função `main()` abaixo e selecione as opções correspondentes à saída impressa do programa.

```
def f(L, x):
    L.append(x)
    i = len(L)-2
    while x < L[i]:
        L[i+1] = L[i]
        i -= 1
    L[i+1] = x

def g(P):
    s = 0
    for i in range(1,len(P)):
        if P[i] > P[i-1]:
            s += 1
    return s
```

```
def main():
    L = [28, 34, 72, 10, 82, 56, 63]
    P = [1, 3, 2, 4, 0, 5]
    P.append(len(P))
    print(L[g(P)])
    temp = P[2]
    P[2] = P[5]
    P[5] = temp
    for i in range(len(P)-5):
        print(L[P[i*2]])
    f(L, 42)
    A = L[2:5]
    print(A[1])
    print(A[2])
```

Prova — questão 2

Implemente uma função chamada `frequencia_relativa(lst)` que recebe como parâmetro uma lista `lst` e devolve duas listas indicando a frequência relativa dos elementos de `lst`. Por exemplo, para a lista `lst = [3, 4, 5, 4, 5, 2, 7, 4, 4, 2]`, a função deve devolver as listas

```
[3, 4, 5, 2, 7], [0.1, 0.4, 0.2, 0.2, 0.1]
```

indicando que 3 corresponde a 10% dos elementos, 4 corresponde a 40% dos elementos e assim por diante.

Para facilitar a implementação de `frequencia_relativa()`, usaremos a função `pertence(n, l)` que verifica se um inteiro `n` ocorre na lista `l`. Em caso afirmativo, a função devolve a posição (índice) da primeira ocorrência; caso contrário, devolve -1.

Prova — questão 2

```
def pertence(n, l):
    for i in range( L1 ):
        if l[i] == L2 :
            return L3
    return -1

def frequencia_relativa(lst):
    distintos = []
    L4
    for elemento in lst:
        L5
        if indice == -1:
            distintos.append(elemento)
            L6
        else:
            frequencia[indice] += 1

    for i in range(len(frequencia)):
        L7
    return L8
```

Prova — questão 2

L1:

- `len(l)`
- `len(i)`
- `l`
- `n`
- `l[i]`

L2:

- `n`
- `l[n]`
- `len(l)`
- `0`
- `l[i]`

L3:

- `i`
- `l`
- `n`
- `+1`
- `l[i]`

L4:

- `frequencia = []`
- `frequencia = 0`
- `distintos.append(frequencia)`
- `frequencia = [[]]`
- `distintos.append(lst)`

Prova — questão 2

L1:

- `len(l)`
- `len(i)`
- `l`
- `n`
- `l[i]`

L2:

- `n`
- `l[n]`
- `len(l)`
- `0`
- `l[i]`

L3:

- `i`
- `l`
- `n`
- `+1`
- `l[i]`

L4:

- `frequencia = []`
- `frequencia = 0`
- `distintos.append(frequencia)`
- `frequencia = [[]]`
- `distintos.append(lst)`

Prova — questão 2

L1:

- `len(l)`
- `len(i)`
- `l`
- `n`
- `l[i]`

L2:

- `n`
- `l[n]`
- `len(l)`
- `0`
- `l[i]`

L3:

- `i`
- `l`
- `n`
- `+1`
- `l[i]`

L4:

- `frequencia = []`
- `frequencia = 0`
- `distintos.append(frequencia)`
- `frequencia = [[]]`
- `distintos.append(lst)`

Prova — questão 2

L1:

- `len(l)`
- `len(i)`
- `l`
- `n`
- `l[i]`

L2:

- `n`
- `l[n]`
- `len(l)`
- `0`
- `l[i]`

L3:

- `i`
- `l`
- `n`
- `+1`
- `l[i]`

L4:

- `frequencia = []`
- `frequencia = 0`
- `distintos.append(frequencia)`
- `frequencia = [[]]`
- `distintos.append(lst)`

Prova — questão 2

L1:

- `len(l)`
- `len(i)`
- `l`
- `n`
- `l[i]`

L2:

- `n`
- `l[n]`
- `len(l)`
- `0`
- `l[i]`

L3:

- `i`
- `l`
- `n`
- `+1`
- `l[i]`

L4:

- `frequencia = []`
- `frequencia = 0`
- `distintos.append(frequencia)`
- `frequencia = [[]]`
- `distintos.append(lst)`

Prova — questão 2

L5:

- `indice = pertence(elemento, distintos)`
- `indice = frequencia_relativa(elemento, distintos)`
- `indice = frequencia_relativa(lst)`
- `indice = pertence(distintos, elemento)`
- `elemento = pertence(distintos, indice)`

L6:

- `frequencia.append(1)`
- `frequencia.append(-1)`
- `frequencia[indice] = 1`
- `elemento.next()`
- `frequencia.append(0)`

L7:

- `frequencia[i] /= len(lst)`
- `frequencia[i] /= lst`
- `frequencia[i] *= len(lst)`
- `frequencia[i] /= 1/len(lst)`
- `frequencia[i] += 1`

L8:

- `distintos, frequencia`
- `elemento, distintos`
- `frequencia, distintos`
- `lst, frequencia`
- `frequencia, elemento`

Prova — questão 2

L5:

- `indice = pertence(elemento, distintos)`
- `indice = frequencia_relativa(elemento, distintos)`
- `indice = frequencia_relativa(lst)`
- `indice = pertence(distintos, elemento)`
- `elemento = pertence(distintos, indice)`

L6:

- `frequencia.append(1)`
- `frequencia.append(-1)`
- `frequencia[indice] = 1`
- `elemento.next()`
- `frequencia.append(0)`

L7:

- `frequencia[i] /= len(lst)`
- `frequencia[i] /= lst`
- `frequencia[i] *= len(lst)`
- `frequencia[i] /= 1/len(lst)`
- `frequencia[i] += 1`

L8:

- `distintos, frequencia`
- `elemento, distintos`
- `frequencia, distintos`
- `lst, frequencia`
- `frequencia, elemento`

Prova — questão 2

L5:

- `indice = pertence(elemento, distintos)`
- `indice = frequencia_relativa(elemento, distintos)`
- `indice = frequencia_relativa(lst)`
- `indice = pertence(distintos, elemento)`
- `elemento = pertence(distintos, indice)`

L6:

- `frequencia.append(1)`
- `frequencia.append(-1)`
- `frequencia[indice] = 1`
- `elemento.next()`
- `frequencia.append(0)`

L7:

- `frequencia[i] /= len(lst)`
- `frequencia[i] /= lst`
- `frequencia[i] *= len(lst)`
- `frequencia[i] /= 1/len(lst)`
- `frequencia[i] += 1`

L8:

- `distintos, frequencia`
- `elemento, distintos`
- `frequencia, distintos`
- `lst, frequencia`
- `frequencia, elemento`

Prova — questão 2

L5:

- `indice = pertence(elemento, distintos)`
- `indice = frequencia_relativa(elemento, distintos)`
- `indice = frequencia_relativa(lst)`
- `indice = pertence(distintos, elemento)`
- `elemento = pertence(distintos, indice)`

L6:

- `frequencia.append(1)`
- `frequencia.append(-1)`
- `frequencia[indice] = 1`
- `elemento.next()`
- `frequencia.append(0)`

L7:

- `frequencia[i] /= len(lst)`
- `frequencia[i] /= lst`
- `frequencia[i] *= len(lst)`
- `frequencia[i] /= 1/len(lst)`
- `frequencia[i] += 1`

L8:

- `distintos, frequencia`
- `elemento, distintos`
- `frequencia, distintos`
- `lst, frequencia`
- `frequencia, elemento`

Prova — questão 2

L5:

- `indice = pertence(elemento, distintos)`
- `indice = frequencia_relativa(elemento, distintos)`
- `indice = frequencia_relativa(lst)`
- `indice = pertence(distintos, elemento)`
- `elemento = pertence(distintos, indice)`

L6:

- `frequencia.append(1)`
- `frequencia.append(-1)`
- `frequencia[indice] = 1`
- `elemento.next()`
- `frequencia.append(0)`

L7:

- `frequencia[i] /= len(lst)`
- `frequencia[i] /= lst`
- `frequencia[i] *= len(lst)`
- `frequencia[i] /= 1/len(lst)`
- `frequencia[i] += 1`

L8:

- `distintos, frequencia`
- `elemento, distintos`
- `frequencia, distintos`
- `lst, frequencia`
- `frequencia, elemento`

Prova — questão 3

Faça uma função `distancia(P, A)` que calcula a distância de uma nave na posição `P = [x, y]` até a superfície de um astro (corpo celeste) `A = [[xc, yc], R]`, onde `xc` e `yc` são a posição do seu centro e `R` é o valor do seu raio. Considere que a nave possui dimensões desprezíveis em relação ao tamanho do astro e que não há colisões.

A trajetória de uma nave pode ser descrita por uma lista de `t` pontos `T = [P0, P1,…, Pt-1]`, com as posições `Pi = [xi, yi]`, $0 \leq$ `i < t` amostradas da nave em intervalos fixos de tempo. Dada a trajetória `T` e um astro `A`, faça uma função `distancia_minima(T, A)` que devolve qual foi a menor distância a que a nave esteve do astro ao longo da sua trajetória.

Prova — questão 3

Dadas as trajetórias de um conjunto de `n` naves na forma de uma lista `LN = [T0, T1,…, Tn-1]`, faça uma função `nave_percorreu_maior_distancia(LN)` que calcula qual nave percorreu a maior distância total e devolve o índice dessa nave e o valor da distância percorrida. Dadas essas mesmas trajetórias e uma lista de `m` astros, faça um programa que diz qual nave percorreu a maior distância e qual foi a distância mínima que cada nave assumiu ao longo de sua trajetória em relação a cada um dos astros.

Exemplo de execução do programa completo

```
Maior percurso: nave 1 (2.41Km)
Menores distâncias:
nave 0 -> astro 0: 233335.03Km
nave 0 -> astro 1: 44000.00Km
nave 1 -> astro 0: 284.27Km
nave 1 -> astro 1: 163845.64Km
```

Prova — questão 3

```
def nave_percorreu_maior_distancia(LN):
    im, dm = 0, 0.0
    for i in L1 :
        d = 0.0
        for j in L2 :
            L3
            L4
            L5a/5b
    return im,dm

def distancia(P, A):
    return L6

def distancia_minima(T, A):
    dm = -1
    for p in L7 :
        L8
        if L9 :
            dm = d
    return dm
```

```
def main():
  T0 = [[150000,214002],[150000,214001],[150000,214000]]
  T1 = [[20000, 20000],[20000, 20001],[20001, 20002]]
  LN = [T0, T1]
  LA = [  [[0,0], 28000],  [[150000,150000], 20000]  ]
  i,d = nave_percorreu_maior_distancia(LN)
  print("Maior percurso: nave {} ({:.2f}Km)".format(i,d))
  print("Menores distâncias:")
  for i in range(len(LN)):
    for j in L10 :
      dm = L11
      print("nave {} -> astro {}: {:.2f}Km".format(i,j,dm))
```

Prova — questão 3

L1:

- `range(len(LN))`
- `len(LN)`
- `rangelen(LN,0,-1)`
- `range(LN)`
- `range(1, len(LN))`

L2:

- `range(1, len(LN[i]))`
- `len(LN[i])`
- `range(LN[i],1,-1)`
- `range(1, LN[i]-1)`
- `range(len(LN[i]))`

L3:

- `P,Pa = LN[i][j],LN[i][j-1]`
- `P,Pa = LN[i][j],LN[i][j+1]`
- `P,Pa = [i,j],[i,j-1]`
- `P,Pa = LN[j][i],LN[j][i+1]`
- `P,Pa = LN[i],LN[j]`

Prova — questão 3

L1:

- `range(len(LN))`
- `len(LN)`
- `rangelen(LN,0,-1)`
- `range(LN)`
- `range(1, len(LN))`

L2:

- `range(1, len(LN[i]))`
- `len(LN[i])`
- `range(LN[i],1,-1)`
- `range(1, LN[i]-1)`
- `range(len(LN[i]))`

L3:

- `P,Pa = LN[i][j],LN[i][j-1]`
- `P,Pa = LN[i][j],LN[i][j+1]`
- `P,Pa = [i,j],[i,j-1]`
- `P,Pa = LN[j][i],LN[j][i+1]`
- `P,Pa = LN[i],LN[j]`

Prova — questão 3

L1:

- `range(len(LN))`
- `len(LN)`
- `rangelen(LN,0,-1)`
- `range(LN)`
- `range(1, len(LN))`

L2:

- `range(1, len(LN[i]))`
- `len(LN[i])`
- `range(LN[i],1,-1)`
- `range(1, LN[i]-1)`
- `range(len(LN[i]))`

L3:

- `P,Pa = LN[i][j],LN[i][j-1]`
- `P,Pa = LN[i][j],LN[i][j+1]`
- `P,Pa = [i,j],[i,j-1]`
- `P,Pa = LN[j][i],LN[j][i+1]`
- `P,Pa = LN[i],LN[j]`

Prova — questão 3

L1:

- `range(len(LN))`
- `len(LN)`
- `rangelen(LN,0,-1)`
- `range(LN)`
- `range(1, len(LN))`

L2:

- `range(1, len(LN[i]))`
- `len(LN[i])`
- `range(LN[i],1,-1)`
- `range(1, LN[i]-1)`
- `range(len(LN[i]))`

L3:

- `P,Pa = LN[i][j],LN[i][j-1]`
- `P,Pa = LN[i][j],LN[i][j+1]`
- `P,Pa = [i,j],[i,j-1]`
- `P,Pa = LN[j][i],LN[j][i+1]`
- `P,Pa = LN[i],LN[j]`

Prova — questão 3

L4:

- `d += ((P[0]-Pa[0])**2 + (P[1]-Pa[1])**2)**0.5`
- `d += ((P[0]-Pa[1])**2 + (P[0]-Pa[1])**2)**0.5`
- `d += ((P[0]-P[1])**2 + (Pa[0]-Pa[1])**2)**0.5`
- `d += (P[0]-Pa[0]) - (P[1]-Pa[1])`
- `d += ((P-Pa)**2)**1/2`

L5a/b:

- ```
  if d > dm:
      im,dm = i,d
  ```
- ```
  if d < dm:
      im,dm = i,d
  ```
- ```
  if d**0.5 < dm:
      dm = d**0.5
  ```
- ```
  if d > dm:
      im = i
  ```
- ```
  if d > dm:
      return i,d
  ```
  ```

Prova — questão 3

L4:

- `d += ((P[0]-Pa[0])**2 + (P[1]-Pa[1])**2)**0.5`
- `d += ((P[0]-Pa[1])**2 + (P[0]-Pa[1])**2)**0.5`
- `d += ((P[0]-P[1])**2 + (Pa[0]-Pa[1])**2)**0.5`
- `d += (P[0]-Pa[0]) - (P[1]-Pa[1])`
- `d += ((P-Pa)**2)**1/2`

L5a/b:

- ```
  if d > dm:
      im,dm = i,d
  ```
- ```
  if d < dm:
      im,dm = i,d
  ```
- ```
  if d**0.5 < dm:
      dm = d**0.5
  ```
- ```
  if d > dm:
      im = i
  ```
- ```
  if d > dm:
      return i,d
  ```
  ```

Prova — questão 3

L4:

- `d += ((P[0]-Pa[0])**2 + (P[1]-Pa[1])**2)**0.5`
- `d += ((P[0]-Pa[1])**2 + (P[0]-Pa[1])**2)**0.5`
- `d += ((P[0]-P[1])**2 + (Pa[0]-Pa[1])**2)**0.5`
- `d += (P[0]-Pa[0]) - (P[1]-Pa[1])`
- `d += ((P-Pa)**2)**1/2`

L5a/b:

- ```
  if d > dm:
      im,dm = i,d
  ```
- ```
  if d < dm:
      im,dm = i,d
  ```
- ```
  if d**0.5 < dm:
      dm = d**0.5
  ```
- ```
  if d > dm:
      im = i
  ```
- ```
  if d > dm:
      return i,d
  ```
  ```

Prova — questão 3

L6:

- `((P[0]-A[0][0])**2 + (P[1]-A[0][1])**2)**0.5-A[1]`
- `((P[0]-A[0][0])**2 + (P[1]-A[0][1])**2)**0.5`
- `((P[0]-A[0][0])**2 + (P[1]-A[1][0])**2)**1/2-A[2]`
- `((P[0]-A[0][0])*2 + (P[1]-A[1][0])*2)**0.5-A[1]`
- `((P[0]-A[0])**2 + (P[1]-A[1])**2)-A[1]`

L7:

- `T`
- `len(T)`
- `range(T)`
- `range(len(T))`
- `range(1,len(T)+1)`

L8:

- `d = distancia(p, A)`
- `d = distancia(T[i], A[0])`
- `d = distancia(p, A[0])`
- `d = distancia(T[i][0], A)`
- `d = distancia(p, A[i][0])`

Prova — questão 3

L6:

- `((P[0]-A[0][0])**2 + (P[1]-A[0][1])**2)**0.5-A[1]`
- `((P[0]-A[0][0])**2 + (P[1]-A[0][1])**2)**0.5`
- `((P[0]-A[0][0])**2 + (P[1]-A[1][0])**2)**1/2-A[2]`
- `((P[0]-A[0][0])*2 + (P[1]-A[1][0])*2)**0.5-A[1]`
- `((P[0]-A[0])**2 + (P[1]-A[1])**2)-A[1]`

L7:

- `T`
- `len(T)`
- `range(T)`
- `range(len(T))`
- `range(1,len(T)+1)`

L8:

- `d = distancia(p, A)`
- `d = distancia(T[i], A[0])`
- `d = distancia(p, A[0])`
- `d = distancia(T[i][0], A)`
- `d = distancia(p, A[i][0])`

Prova — questão 3

L6:

- `((P[0]-A[0][0])**2 + (P[1]-A[0][1])**2)**0.5-A[1]`
- `((P[0]-A[0][0])**2 + (P[1]-A[0][1])**2)**0.5`
- `((P[0]-A[0][0])**2 + (P[1]-A[1][0])**2)**1/2-A[2]`
- `((P[0]-A[0][0])*2 + (P[1]-A[1][0])*2)**0.5-A[1]`
- `((P[0]-A[0])**2 + (P[1]-A[1])**2)-A[1]`

L7:

- `T`
- `len(T)`
- `range(T)`
- `range(len(T))`
- `range(1,len(T)+1)`

L8:

- `d = distancia(p, A)`
- `d = distancia(T[i], A[0])`
- `d = distancia(p, A[0])`
- `d = distancia(T[i][0], A)`
- `d = distancia(p, A[i][0])`

Prova — questão 3

L6:

- `((P[0]-A[0][0])**2 + (P[1]-A[0][1])**2)**0.5-A[1]`
- `((P[0]-A[0][0])**2 + (P[1]-A[0][1])**2)**0.5`
- `((P[0]-A[0][0])**2 + (P[1]-A[1][0])**2)**1/2-A[2]`
- `((P[0]-A[0][0])*2 + (P[1]-A[1][0])*2)**0.5-A[1]`
- `((P[0]-A[0])**2 + (P[1]-A[1])**2)-A[1]`

L7:

- `T`
- `len(T)`
- `range(T)`
- `range(len(T))`
- `range(1,len(T)+1)`

L8:

- `d = distancia(p, A)`
- `d = distancia(T[i], A[0])`
- `d = distancia(p, A[0])`
- `d = distancia(T[i][0], A)`
- `d = distancia(p, A[i][0])`

Prova — questão 3

L9:

- `dm == -1 or d < dm`
- `dm == -1 and d < dm`
- `dm != -1 or d-dm > 0`
- `dm == -1 or d > dm`
- `not(dm == -1 and d < dm)`

L10:

- `range(len(LA))`
- `range(len(LA)-1)`
- `range(1,len(LA))`
- `range(len(LA)-1,0,-1)`
- `range(LA)`

L11:

- `distancia_minima(LN[i], LA[j])`
- `distancia(LN[i], LA[j])`
- `distancia_minima(LN[j], LA[i])`
- `distancia(LN[i][j], LA[j][0])`
- `distancia_minima(LN[i], LA[j][0])`

Prova — questão 3

L9:

- `dm == -1 or d < dm`
- `dm == -1 and d < dm`
- `dm != -1 or d-dm > 0`
- `dm == -1 or d > dm`
- `not(dm == -1 and d < dm)`

L10:

- `range(len(LA))`
- `range(len(LA)-1)`
- `range(1,len(LA))`
- `range(len(LA)-1,0,-1)`
- `range(LA)`

L11:

- `distancia_minima(LN[i], LA[j])`
- `distancia(LN[i], LA[j])`
- `distancia_minima(LN[j], LA[i])`
- `distancia(LN[i][j], LA[j][0])`
- `distancia_minima(LN[i], LA[j][0])`

Prova — questão 3

L9:

- `dm == -1 or d < dm`
- `dm == -1 and d < dm`
- `dm != -1 or d-dm > 0`
- `dm == -1 or d > dm`
- `not(dm == -1 and d < dm)`

L10:

- `range(len(LA))`
- `range(len(LA)-1)`
- `range(1,len(LA))`
- `range(len(LA)-1,0,-1)`
- `range(LA)`

L11:

- `distancia_minima(LN[i], LA[j])`
- `distancia(LN[i], LA[j])`
- `distancia_minima(LN[j], LA[i])`
- `distancia(LN[i][j], LA[j][0])`
- `distancia_minima(LN[i], LA[j][0])`

Prova — questão 3

L9:

- `dm == -1 or d < dm`
- `dm == -1 and d < dm`
- `dm != -1 or d-dm > 0`
- `dm == -1 or d > dm`
- `not(dm == -1 and d < dm)`

L10:

- `range(len(LA))`
- `range(len(LA)-1)`
- `range(1,len(LA))`
- `range(len(LA)-1,0,-1)`
- `range(LA)`

L11:

- `distancia_minima(LN[i], LA[j])`
- `distancia(LN[i], LA[j])`
- `distancia_minima(LN[j], LA[i])`
- `distancia(LN[i][j], LA[j][0])`
- `distancia_minima(LN[i], LA[j][0])`

Prova — questão 4

Uma matriz M é dita **Matriz de Hadamard** se (1) M é quadrada; (2) todas as suas células contêm 1 ou -1; e (3) todas as suas linhas são ortogonais entre si, ou seja, o produto escalar de quaisquer duas linhas é 0. Lembre-se que o produto escalar de dois vetores $(a_1, \ldots, a_n)$ e $(b_1, \ldots, b_n)$ é o somatório $\sum_{i=1}^{n} a_i * b_i = a_1 \cdot b_1 + a_2 \cdot b_2 + \cdots + a_n \cdot b_n$.

Crie a função `matriz_hadamard(M)`, que decide se uma dada matriz é uma Matriz de Hadamard. Para isso, vamos usar a função auxiliar `ortogonais(l1, l2)`, que decide se duas linhas dadas são ortogonais.

Prova — questão 4

```
def ortogonais(linha1, linha2):
    produto_escalar = 0
    for i in range(len(linha1)):
        produto_escalar += linha1[i]* L1
    return L2

def matriz_hadamard(M):
    n,m = len(M), L3

    if n != m:
        L4

    L5
    for i in range(n):
        for j in L6:

            if L7:
                L8

            if i != j and L9:
                é_hadamard = False

    return L10
```

Prova — questão 4

L1:

- `linha2[i]`
- `linha2[j]`
- `2`
- `len(linha1)`
- `i`

L2:

- `(produto_escalar == 0)`
- `True`
- `False`
- `produto_escalar`
- `produto_escalar < 0`

L3:

- `len(M[0])`
- `n`
- `M[0]`
- `len(M)`
- `M[0][0]`

L4:

- `return False`
- `return True`
- `m = n`
- `m += 1`
- `return n != m`

Prova — questão 4

L1:

- `linha2[i]`
- `linha2[j]`
- `2`
- `len(linha1)`
- `i`

L2:

- `(produto_escalar == 0)`
- `True`
- `False`
- `produto_escalar`
- `produto_escalar < 0`

L3:

- `len(M[0])`
- `n`
- `M[0]`
- `len(M)`
- `M[0][0]`

L4:

- `return False`
- `return True`
- `m = n`
- `m += 1`
- `return n != m`

Prova — questão 4

L1:

- `linha2[i]`
- `linha2[j]`
- `2`
- `len(linha1)`
- `i`

L2:

- `(produto_escalar == 0)`
- `True`
- `False`
- `produto_escalar`
- `produto_escalar < 0`

L3:

- `len(M[0])`
- `n`
- `M[0]`
- `len(M)`
- `M[0][0]`

L4:

- `return False`
- `return True`
- `m = n`
- `m += 1`
- `return n != m`

Prova — questão 4

L1:

- `linha2[i]`
- `linha2[j]`
- `2`
- `len(linha1)`
- `i`

L2:

- `(produto_escalar == 0)`
- `True`
- `False`
- `produto_escalar`
- `produto_escalar < 0`

L3:

- `len(M[0])`
- `n`
- `M[0]`
- `len(M)`
- `M[0][0]`

L4:

- `return False`
- `return True`
- `m = n`
- `m += 1`
- `return n != m`

Prova — questão 4

L1:

- `linha2[i]`
- `linha2[j]`
- `2`
- `len(linha1)`
- `i`

L2:

- `(produto_escalar == 0)`
- `True`
- `False`
- `produto_escalar`
- `produto_escalar < 0`

L3:

- `len(M[0])`
- `n`
- `M[0]`
- `len(M)`
- `M[0][0]`

L4:

- `return False`
- `return True`
- `m = n`
- `m += 1`
- `return n != m`

Prova — questão 4

L5:

- `é_hadamard = True`
- `é_hadamard = False`
- `i = 0`
- `n = m`
- `i,j = 0,0`

L6:

- `range(n)`
- `range(n+1)`
- `range(i+1)`
- `range(2*i)`
- `M[i]`

L7:

- `M[i][j] != 1 and M[i][j] != -1`
- `M[i][j] == 1 and M[i][j] == -1`
- `M[i][j] != 1 or M[i][j] != -1`
- `M[i][j] == 1 or M[i][j] == -1`
- `M[i][j] == -1 or M[i][j] == 1`

L8:

- `é_hadamard = False`
- `é_hadamard = True`
- `é_hadamard = not é_hadamard`
- `i += 1`
- `j += 1`

Prova — questão 4

L5:

- `é_hadamard = True`
- `é_hadamard = False`
- `i = 0`
- `n = m`
- `i,j = 0,0`

L6:

- `range(n)`
- `range(n+1)`
- `range(i+1)`
- `range(2*i)`
- `M[i]`

L7:

- `M[i][j] != 1 and M[i][j] != -1`
- `M[i][j] == 1 and M[i][j] == -1`
- `M[i][j] != 1 or M[i][j] != -1`
- `M[i][j] == 1 or M[i][j] == -1`
- `M[i][j] == -1 or M[i][j] == 1`

L8:

- `é_hadamard = False`
- `é_hadamard = True`
- `é_hadamard = not é_hadamard`
- `i += 1`
- `j += 1`

Prova — questão 4

L5:

- `é_hadamard = True`
- `é_hadamard = False`
- `i = 0`
- `n = m`
- `i,j = 0,0`

L6:

- `range(n)`
- `range(n+1)`
- `range(i+1)`
- `range(2*i)`
- `M[i]`

L7:

- `M[i][j] != 1 and M[i][j] != -1`
- `M[i][j] == 1 and M[i][j] == -1`
- `M[i][j] != 1 or M[i][j] != -1`
- `M[i][j] == 1 or M[i][j] == -1`
- `M[i][j] == -1 or M[i][j] == 1`

L8:

- `é_hadamard = False`
- `é_hadamard = True`
- `é_hadamard = not é_hadamard`
- `i += 1`
- `j += 1`

Prova — questão 4

L5:

- `é_hadamard = True`
- `é_hadamard = False`
- `i = 0`
- `n = m`
- `i,j = 0,0`

L6:

- `range(n)`
- `range(n+1)`
- `range(i+1)`
- `range(2*i)`
- `M[i]`

L7:

- `M[i][j] != 1 and M[i][j] != -1`
- `M[i][j] == 1 and M[i][j] == -1`
- `M[i][j] != 1 or M[i][j] != -1`
- `M[i][j] == 1 or M[i][j] == -1`
- `M[i][j] == -1 or M[i][j] == 1`

L8:

- `é_hadamard = False`
- `é_hadamard = True`
- `é_hadamard = not é_hadamard`
- `i += 1`
- `j += 1`

Prova — questão 4

L5:

- **`é_hadamard = True`**
- `é_hadamard = False`
- `i = 0`
- `n = m`
- `i,j = 0,0`

L6:

- **`range(n)`**
- `range(n+1)`
- `range(i+1)`
- `range(2*i)`
- `M[i]`

L7:

- **`M[i][j] != 1 and M[i][j] != -1`**
- `M[i][j] == 1 and M[i][j] == -1`
- `M[i][j] != 1 or M[i][j] != -1`
- `M[i][j] == 1 or M[i][j] == -1`
- `M[i][j] == -1 or M[i][j] == 1`

L8:

- **`é_hadamard = False`**
- `é_hadamard = True`
- `é_hadamard = not é_hadamard`
- `i += 1`
- `j += 1`

Prova — questão 4

L9:

- `not ortogonais(M[i],M[j])`
- `ortogonais(not M[i],not M[j])`
- `not ortogonais(M[i],M[i+j])`
- `ortogonais(M[i],M[i+j])`
- `ortogonais(M[i],M[j])`

L10:

- `é_hadamard`
- `not é_hadamard`
- `True`
- `False`
- `é_hadamard == False`

Prova — questão 4

L9:

- `not ortogonais(M[i],M[j])`
- `ortogonais(not M[i],not M[j])`
- `not ortogonais(M[i],M[i+j])`
- `ortogonais(M[i],M[i+j])`
- `ortogonais(M[i],M[j])`

L10:

- `é_hadamard`
- `not é_hadamard`
- `True`
- `False`
- `é_hadamard == False`

Prova — questão 4

L9:

- `not ortogonais(M[i],M[j])`
- `ortogonais(not M[i],not M[j])`
- `not ortogonais(M[i],M[i+j])`
- `ortogonais(M[i],M[i+j])`
- `ortogonais(M[i],M[j])`

L10:

- `é_hadamard`
- `not é_hadamard`
- `True`
- `False`
- `é_hadamard == False`

Prova — questão 5

- **permitem quebrar tarefas grandes em menores, facilitando o entendimento e a manutenção do código**
- **permitem reaproveitar código existente, minimizando a duplicação de código**
- **permitem esconder detalhes de implementação dentro de um trecho de código, promovendo uma visão de mais alto nível que viabiliza programas mais complexos**
- **permitem resolver alguns problemas que seriam impossíveis de resolver computacionalmente sem funções**
- **permitem implementar soluções mais eficientes, que chegam na solução mais rapidamente**
- **permitem o uso de depuradores (*debuggers*), que facilitam a correção de erros.**
- **permitem que o código fique mais extenso, aproveitando melhor a memória disponível**

Prova — questão 5

- **permitem quebrar tarefas grandes em menores, facilitando o entendimento e a manutenção do código**
- **permitem reaproveitar código existente, minimizando a duplicação de código**
- **permitem esconder detalhes de implementação dentro de um trecho de código, promovendo uma visão de mais alto nível que viabiliza programas mais complexos**
- **permitem resolver alguns problemas que seriam impossíveis de resolver computacionalmente sem funções**
- **permitem implementar soluções mais eficientes, que chegam na solução mais rapidamente**
- **permitem o uso de depuradores (*debuggers*), que facilitam a correção de erros.**
- **permitem que o código fique mais extenso, aproveitando melhor a memória disponível**

And now for something completely different

Módulos

- Usamos funções para nos auxiliar a enxergar o programa (e o problema) em partes de tamanho “palatável”

Módulos

- Usamos funções para nos auxiliar a enxergar o programa (e o problema) em partes de tamanho "palatável"
- Quando o programa fica grande, queremos fazer o mesmo em um nível mais alto

Módulos

- Usamos funções para nos auxiliar a enxergar o programa (e o problema) em partes de tamanho “palatável”
- Quando o programa fica grande, queremos fazer o mesmo em um nível mais alto
- O que fazemos é dividir o programa em arquivos diferentes que agrupam as funções “por assunto”

Módulos

- **Usamos funções para nos auxiliar a enxergar o programa (e o problema) em partes de tamanho "palatável"**
- **Quando o programa fica grande, queremos fazer o mesmo em um nível mais alto**
- **O que fazemos é dividir o programa em arquivos diferentes que agrupam as funções "por assunto"**
  - Por exemplo, em uma loja online:

Módulos

- **Usamos funções para nos auxiliar a enxergar o programa (e o problema) em partes de tamanho "palatável"**
- **Quando o programa fica grande, queremos fazer o mesmo em um nível mais alto**
- **O que fazemos é dividir o programa em arquivos diferentes que agrupam as funções "por assunto"**
  - Por exemplo, em uma loja online:
    - » *Funções relacionadas à busca por produtos e apresentação de produtos similares*

Módulos

- **Usamos funções para nos auxiliar a enxergar o programa (e o problema) em partes de tamanho "palatável"**
- **Quando o programa fica grande, queremos fazer o mesmo em um nível mais alto**
- **O que fazemos é dividir o programa em arquivos diferentes que agrupam as funções "por assunto"**
  - Por exemplo, em uma loja online:
    - » *Funções relacionadas à busca por produtos e apresentação de produtos similares*
    - » *Funções relacionadas ao "carrinho"*

Módulos

- **Usamos funções para nos auxiliar a enxergar o programa (e o problema) em partes de tamanho "palatável"**
- **Quando o programa fica grande, queremos fazer o mesmo em um nível mais alto**
- **O que fazemos é dividir o programa em arquivos diferentes que agrupam as funções "por assunto"**
  - Por exemplo, em uma loja online:
    - » *Funções relacionadas à busca por produtos e apresentação de produtos similares*
    - » *Funções relacionadas ao "carrinho"*
    - » *Funções relacionadas ao processo de pagamento*

Módulos

- **Usamos funções para nos auxiliar a enxergar o programa (e o problema) em partes de tamanho "palatável"**
- **Quando o programa fica grande, queremos fazer o mesmo em um nível mais alto**
- **O que fazemos é dividir o programa em arquivos diferentes que agrupam as funções "por assunto"**
  - Por exemplo, em uma loja online:
    - » *Funções relacionadas à busca por produtos e apresentação de produtos similares*
    - » *Funções relacionadas ao "carrinho"*
    - » *Funções relacionadas ao processo de pagamento*
    - » *Funções relacionadas à inserção e remoção de produtos no catálogo do sítio*

Módulos

- **Usamos funções para nos auxiliar a enxergar o programa (e o problema) em partes de tamanho "palatável"**
- **Quando o programa fica grande, queremos fazer o mesmo em um nível mais alto**
- **O que fazemos é dividir o programa em arquivos diferentes que agrupam as funções "por assunto"**
  - Por exemplo, em uma loja online:
    - » *Funções relacionadas à busca por produtos e apresentação de produtos similares*
    - » *Funções relacionadas ao "carrinho"*
    - » *Funções relacionadas ao processo de pagamento*
    - » *Funções relacionadas à inserção e remoção de produtos no catálogo do sítio*
    - » *...*

Módulos

- **Além de ajudar na organização do programa, isso tem outras vantagens**

Módulos

- **Além de ajudar na organização do programa, isso tem outras vantagens**
  - Fica mais fácil integrar o trabalho de diferentes pessoas

Módulos

- **Além de ajudar na organização do programa, isso tem outras vantagens**
  - Fica mais fácil integrar o trabalho de diferentes pessoas
  - É possível reutilizar as funções de um arquivo em outros programas

Módulos

- **Além de ajudar na organização do programa, isso tem outras vantagens**
  - Fica mais fácil integrar o trabalho de diferentes pessoas
  - É possível reutilizar as funções de um arquivo em outros programas
    - » *Por exemplo, as funções relacionadas ao sistema de pagamento da loja online podem ser usadas também para o sistema do caixa nas lojas físicas*

Módulos

- **Além de ajudar na organização do programa, isso tem outras vantagens**
  - Fica mais fácil integrar o trabalho de diferentes pessoas
  - É possível reutilizar as funções de um arquivo em outros programas
    - » *Por exemplo, as funções relacionadas ao sistema de pagamento da loja online podem ser usadas também para o sistema do caixa nas lojas físicas*
  - É possível publicar arquivos com funções úteis para que sejam usados por outras pessoas (e você também pode usar arquivos criados por outras pessoas)

Módulos

- MAS!

Módulos

- MAS!
- Qual a chance de duas pessoas em situações diferentes criarem duas funções com o mesmo nome?

Módulos

- MAS!
- Qual a chance de duas pessoas em situações diferentes criarem duas funções com o mesmo nome?
- Muito alta

Módulos

- MAS!
- Qual a chance de duas pessoas em situações diferentes criarem duas funções com o mesmo nome?
- Muito alta
- A solução para não haver problemas é a mesma que usamos com funções: “cada um com seu cada um e ninguém se mete no cada um dos outros”

Módulos

- **MAS!**
- **Qual a chance de duas pessoas em situações diferentes criarem duas funções com o mesmo nome?**
- **Muito alta**
- **A solução para não haver problemas é a mesma que usamos com funções: "cada um com seu cada um e ninguém se mete no cada um dos outros"**
  - (ou seja, cada arquivo tem seu próprio **escopo**)

Módulos

- Mas... mas... mas...

Módulos

- Mas... mas... mas...
- Com funções, o conteúdo das funções tem um escopo próprio, mas podemos acessar o nome da função normalmente

Módulos

- Mas... mas... mas...
- Com funções, o **conteúdo** das funções tem um escopo próprio, mas podemos acessar o **nome** da função normalmente
- Se queremos “esconder” os nomes das funções de um módulo, como é que vamos poder usá-las? 🤔

Módulos

- Mas... mas... mas...
- Com funções, o **conteúdo** das funções tem um escopo próprio, mas podemos acessar o **nome** da função normalmente
- Se queremos "esconder" os nomes das funções de um módulo, como é que vamos poder usá-las? 🤔
- Usamos o nome do módulo como um **prefixo** para o nome das funções

Módulos

- Carregamos um módulo com o comando `import`

Módulos

- **Carregamos um módulo com o comando `import`**
  - já vimos `import math` para carregar funções matemáticas pré-prontas

Módulos

- **Carregamos um módulo com o comando `import`**
  - já vimos `import math` para carregar funções matemáticas pré-prontas
- **Ao executar `import blah`, python procura por um arquivo chamado `blah.py` e executa seu conteúdo**

Módulos

- **Carregamos um módulo com o comando `import`**
  - já vimos `import math` para carregar funções matemáticas pré-prontas
- **Ao executar `import blah`, python procura por um arquivo chamado `blah.py` e executa seu conteúdo**
  - ou seja, em "algum lugar" existe um arquivo chamado `math.py` que contém as definições de funções matemáticas como `sqrt()`

Módulos

- **Carregamos um módulo com o comando `import`**
  - já vimos `import math` para carregar funções matemáticas pré-prontas
- **Ao executar `import blah`, python procura por um arquivo chamado `blah.py` e executa seu conteúdo**
  - ou seja, em "algum lugar" existe um arquivo chamado `math.py` que contém as definições de funções matemáticas como `sqrt()`
  - quando fazemos `import math`, esse arquivo é executado e as definições das funções dele ficam disponíveis

Módulos

- **Carregamos um módulo com o comando `import`**
  - já vimos `import math` para carregar funções matemáticas pré-prontas
- **Ao executar `import blah`, python procura por um arquivo chamado `blah.py` e executa seu conteúdo**
  - ou seja, em "algum lugar" existe um arquivo chamado `math.py` que contém as definições de funções matemáticas como `sqrt()`
  - quando fazemos `import math`, esse arquivo é executado e as definições das funções dele ficam disponíveis
  - como o módulo se chama "math", todas as funções definidas dentro dele ficam acessíveis com o prefixo "math." ⟶ "`math.sqrt()`"

Módulos

- Em geral, não há nada de especial com o nome do módulo

Módulos

- **Em geral, não há nada de especial com o nome do módulo**
  - ele é simplesmente o nome do arquivo

Módulos

- **Em geral, não há nada de especial com o nome do módulo**
  - ele é simplesmente o nome do arquivo
    - » *Se você renomear o arquivo, vai precisar modificar o comando* `import` *e todas as chamadas de funções que usam o módulo para usar o nome novo*

Módulos

- **Em geral, não há nada de especial com o nome do módulo**
  - ele é simplesmente o nome do arquivo
    - » *Se você renomear o arquivo, vai precisar modificar o comando* `import` *e todas as chamadas de funções que usam o módulo para usar o nome novo*
- **Também não há nada de especial com o código dentro de um módulo**

Módulos

- **Em geral, não há nada de especial com o nome do módulo**
  - ele é simplesmente o nome do arquivo
    - » *Se você renomear o arquivo, vai precisar modificar o comando* `import` *e todas as chamadas de funções que usam o módulo para usar o nome novo*
- **Também não há nada de especial com o código dentro de um módulo**
  - É um trecho de programa python como outro qualquer

Módulos

- **Em geral, não há nada de especial com o nome do módulo**
  - ele é simplesmente o nome do arquivo
    - » *Se você renomear o arquivo, vai precisar modificar o comando* `import` *e todas as chamadas de funções que usam o módulo para usar o nome novo*
- **Também não há nada de especial com o código dentro de um módulo**
  - É um trecho de programa python como outro qualquer
- **Em particular, é possível pegar um programa existente e carregá-lo como um módulo em outro programa!**

Módulos

- **Porém...**

Módulos

- **Porém...**
- **Um programa carregado como módulo provavelmente vai fazer coisas que não fazem sentido no novo contexto**

Módulos

- **Porém...**
- **Um programa carregado como módulo provavelmente vai fazer coisas que não fazem sentido no novo contexto**
  - Em geral, queremos carregar apenas as definições de funções, não efetivamente executar tudo do programa

Módulos

- **Porém...**
- **Um programa carregado como módulo provavelmente vai fazer coisas que não fazem sentido no novo contexto**
  - Em geral, queremos carregar apenas as definições de funções, não efetivamente executar tudo do programa
- **Seria bom se pudéssemos escolher o que rodar quando o programa é carregado como um módulo**

Módulos

- **Porém...**
- **Um programa carregado como módulo provavelmente vai fazer coisas que não fazem sentido no novo contexto**
  - Em geral, queremos carregar apenas as definições de funções, não efetivamente executar tudo do programa
- **Seria bom se pudéssemos escolher o que rodar quando o programa é carregado como um módulo**
  - "só vou executar esta parte aqui quando eu não for um módulo"

Módulos

- python faz isso ser fácil: quando um módulo é carregado, ele "enxerga" o conteúdo da variável `__name__` como o nome do módulo

Módulos

- **python faz isso ser fácil: quando um módulo é carregado, ele "enxerga" o conteúdo da variável `__name__` como o nome do módulo**
  - (ou seja, o nome do arquivo que foi usado para carregar o módulo sem a extensão `.py`)

Módulos

- **python faz isso ser fácil: quando um módulo é carregado, ele "enxerga" o conteúdo da variável `__name__` como o nome do módulo**
  - (ou seja, o nome do arquivo que foi usado para carregar o módulo sem a extensão `.py`)
- **Fora de qualquer módulo, a variável `__name__` contém a string "`__main__`"**

Módulos

- **python faz isso ser fácil: quando um módulo é carregado, ele "enxerga" o conteúdo da variável `__name__` como o nome do módulo**
  - (ou seja, o nome do arquivo que foi usado para carregar o módulo sem a extensão `.py`)
- **Fora de qualquer módulo, a variável `__name__` contém a string "`__main__`"**
- **Por isso, um "truque" comum é fazer**

```
if __name__ == "__main__":
    main()
```

Módulos

- **python faz isso ser fácil: quando um módulo é carregado, ele "enxerga" o conteúdo da variável `__name__` como o nome do módulo**
  - (ou seja, o nome do arquivo que foi usado para carregar o módulo sem a extensão `.py`)
- **Fora de qualquer módulo, a variável `__name__` contém a string "`__main__`"**
- **Por isso, um "truque" comum é fazer**

```
if __name__ == "__main__":
    main()
```

- **E essa é outra razão pela qual é uma boa ideia sempre usar uma função `main()`** 🙂

Testes

Testes de software

- **Herrar é umano!**

Testes de software

- **Herrar é umano!**
  - E ao programar somos todos muito, muito umanos! 🙃

Testes de software

- **Herrar é umano!**
  - E ao programar somos todos muito, muito umanos! 🙃
- **Alguns bugs (defeitos) são irrelevantes, outros são ligeiramente incômodos, mas alguns podem ser catastróficos**

Testes de software

- **Herrar é umano!**
  - E ao programar somos todos muito, muito umanos! 🙃
- **Alguns bugs (defeitos) são irrelevantes, outros são ligeiramente incômodos, mas alguns podem ser catastróficos**
- **Quando um programa "cresce", bugs que antes não causavam problemas podem passar a causar**

Testes de software

- **Herrar é umano!**
  - E ao programar somos todos muito, muito umanos! 🙃
- **Alguns bugs (defeitos) são irrelevantes, outros são ligeiramente incômodos, mas alguns podem ser catastróficos**
- **Quando um programa "cresce", bugs que antes não causavam problemas podem passar a causar**
  - E, naturalmente, programas maiores tendem a ter mais bugs também

Testes de software

- **Herrar é umano!**
  - E ao programar somos todos muito, muito umanos! 🙃
- **Alguns bugs (defeitos) são irrelevantes, outros são ligeiramente incômodos, mas alguns podem ser catastróficos**
- **Quando um programa "cresce", bugs que antes não causavam problemas podem passar a causar**
  - E, naturalmente, programas maiores tendem a ter mais bugs também

Não existe solução definitiva para esse problema

Testes de software

- **Herrar é umano!**
  - E ao programar somos todos muito, muito umanos! 🙃
- **Alguns bugs (defeitos) são irrelevantes, outros são ligeiramente incômodos, mas alguns podem ser catastróficos**
- **Quando um programa "cresce", bugs que antes não causavam problemas podem passar a causar**
  - E, naturalmente, programas maiores tendem a ter mais bugs também

Não existe solução definitiva para esse problema

Mas podemos minimizá-lo com **testes**

Testes de software

- **Erros de sintaxe**

Testes de software

- **Erros de sintaxe**
  - Em geral, fáceis de detectar durante o desenvolvimento

Testes de software

- **Erros de sintaxe**
  - Em geral, fáceis de detectar durante o desenvolvimento
- **Erros de lógica**

Testes de software

- **Erros de sintaxe**
  - Em geral, fáceis de detectar durante o desenvolvimento
- **Erros de lógica**
  - Alguns acontecem sempre e, portanto, também são fáceis de detectar durante o desenvolvimento

Testes de software

- **Erros de sintaxe**
  - Em geral, fáceis de detectar durante o desenvolvimento
- **Erros de lógica**
  - Alguns acontecem sempre e, portanto, também são fáceis de detectar durante o desenvolvimento
  - Alguns aparecem apenas em casos incomuns

Testes de software

- **Erros de sintaxe**
  - Em geral, fáceis de detectar durante o desenvolvimento
- **Erros de lógica**
  - Alguns acontecem sempre e, portanto, também são fáceis de detectar durante o desenvolvimento
  - Alguns aparecem apenas em casos incomuns
  - Alguns são difíceis de encontrar (*heisenbugs*)

Testes de software

- **Erros de sintaxe**
  - Em geral, fáceis de detectar durante o desenvolvimento
- **Erros de lógica**
  - Alguns acontecem sempre e, portanto, também são fáceis de detectar durante o desenvolvimento
  - Alguns aparecem apenas em casos incomuns
  - Alguns são difíceis de encontrar (*heisenbugs*)
- **Em geral, bugs que não aparecem sempre dependem do dado que está sendo processado**

Testes de software

- **Erros de sintaxe**
  - Em geral, fáceis de detectar durante o desenvolvimento
- **Erros de lógica**
  - Alguns acontecem sempre e, portanto, também são fáceis de detectar durante o desenvolvimento
  - Alguns aparecem apenas em casos incomuns
  - Alguns são difíceis de encontrar (*heisenbugs*)
- **Em geral, bugs que não aparecem sempre dependem do dado que está sendo processado**
  - Por exemplo, alguns valores de entrada causam uma divisão por zero

Testes de software

- **Erros de sintaxe**
  - Em geral, fáceis de detectar durante o desenvolvimento
- **Erros de lógica**
  - Alguns acontecem sempre e, portanto, também são fáceis de detectar durante o desenvolvimento
  - Alguns aparecem apenas em casos incomuns
  - Alguns são difíceis de encontrar (*heisenbugs*)
- **Em geral, bugs que não aparecem sempre dependem do dado que está sendo processado**
  - Por exemplo, alguns valores de entrada causam uma divisão por zero
- **Testar, portanto, envolve executar o código com uma boa variedade de "situações" (dados de entrada e contexto)**

Testes de software

- **Erros de sintaxe**
  - Em geral, fáceis de detectar durante o desenvolvimento
- **Erros de lógica**
  - Alguns acontecem sempre e, portanto, também são fáceis de detectar durante o desenvolvimento
  - Alguns aparecem apenas em casos incomuns
  - Alguns são difíceis de encontrar (*heisenbugs*)
- **Em geral, bugs que não aparecem sempre dependem do dado que está sendo processado**
  - Por exemplo, alguns valores de entrada causam uma divisão por zero
- **Testar, portanto, envolve executar o código com uma boa variedade de "situações" (dados de entrada e contexto)**
  - Em particular, é bom pensar se os testes fazem seu programa passar por todos os trechos do código

Testes de software

- **Testes manuais**

Testes de software

- **Testes manuais**
  - Podem ser úteis em alguns momentos exploratórios (“o que será que acontece se eu fizer assim?”)

Testes de software

- **Testes manuais**
  - Podem ser úteis em alguns momentos exploratórios (“o que será que acontece se eu fizer assim?”)
  - Cansativos e tomam muito tempo

Testes de software

- **Testes manuais**
  - Podem ser úteis em alguns momentos exploratórios (“o que será que acontece se eu fizer assim?”)
  - Cansativos e tomam muito tempo
  - Geralmente feitos de maneira *ad hoc*, então provavelmente vão abranger poucos casos

Testes de software

- **Testes manuais**
  - Podem ser úteis em alguns momentos exploratórios (“o que será que acontece se eu fizer assim?”)
  - Cansativos e tomam muito tempo
  - Geralmente feitos de maneira *ad hoc*, então provavelmente vão abranger poucos casos
  - Na prática, são executados só às vezes

Testes de software

- **Testes manuais**
  - Podem ser úteis em alguns momentos exploratórios ("o que será que acontece se eu fizer assim?")
  - Cansativos e tomam muito tempo
  - Geralmente feitos de maneira *ad hoc*, então provavelmente vão abranger poucos casos
  - Na prática, são executados só às vezes
    - » *Mas se eu testei e funcionou, para que vou querer testar de novo?!*

Testes de software

- **Testes manuais**
  - Podem ser úteis em alguns momentos exploratórios (“o que será que acontece se eu fizer assim?”)
  - Cansativos e tomam muito tempo
  - Geralmente feitos de maneira *ad hoc*, então provavelmente vão abranger poucos casos
  - Na prática, são executados só às vezes
    - » *Mas se eu testei e funcionou, para que vou querer testar de novo?!*
    - » *Testes evitam surpresas quando o programa é modificado (ou seja, todos os dias 😂)*

Testes de software – Histórico

Até o início dos anos 2000, a prática comum era

Testes de software – Histórico

Até o início dos anos 2000, a prática comum era

1. **Escreve o programa "inteiro" (ou quase)**

Testes de software – Histórico

Até o início dos anos 2000, a prática comum era

1. **Escreve o programa "inteiro" (ou quase)**
2. **Executa testes manualmente**

Testes de software – Histórico

Até o início dos anos 2000, a prática comum era

1. **Escreve o programa "inteiro" (ou quase)**
2. **Executa testes manualmente**
   - Diversas empresas tinham uma equipe dedicada a isso

Testes de software – Histórico

Até o início dos anos 2000, a prática comum era

1. **Escreve o programa "inteiro" (ou quase)**
2. **Executa testes manualmente**
   - Diversas empresas tinham uma equipe dedicada a isso
   - Cada bug é descrito em detalhes

Testes de software – Histórico

Até o início dos anos 2000, a prática comum era

1. **Escreve o programa "inteiro" (ou quase)**
2. **Executa testes manualmente**
   - Diversas empresas tinham uma equipe dedicada a isso
   - Cada bug é descrito em detalhes
     - » *Em que circunstâncias o bug aparece (qual operação, quais dados de entrada...)*

Testes de software – Histórico

Até o início dos anos 2000, a prática comum era

1. **Escreve o programa "inteiro" (ou quase)**
2. **Executa testes manualmente**
   - Diversas empresas tinham uma equipe dedicada a isso
   - Cada bug é descrito em detalhes
     - » *Em que circunstâncias o bug aparece (qual operação, quais dados de entrada...)*
     - » *Qual a consequência do bug (resultado inválido, programa trava, inconsistência na interface...)*

Testes de software – Histórico

Até o início dos anos 2000, a prática comum era

1. **Escreve o programa "inteiro" (ou quase)**
2. **Executa testes manualmente**
   - Diversas empresas tinham uma equipe dedicada a isso
   - Cada bug é descrito em detalhes
     - » *Em que circunstâncias o bug aparece (qual operação, quais dados de entrada...)*
     - » *Qual a consequência do bug (resultado inválido, programa trava, inconsistência na interface...)*
3. **Usa o depurador (*debugger*) para encontrar e corrigir as causas dos erros**

Testes de software – Histórico

- Demorado e fortemente dependente do depurador

Testes de software – Histórico

- **Demorado e fortemente dependente do depurador**
  - Como os erros podem estar em qualquer lugar, é preciso usar o depurador para encontrá-los, o que é trabalhoso

Testes de software – Histórico

- **Demorado e fortemente dependente do depurador**
  - Como os erros podem estar em qualquer lugar, é preciso usar o depurador para encontrá-los, o que é trabalhoso
  - Muitos erros de uma vez fazem ser mais difícil resolver cada um

Testes de software – Histórico

- **Demorado e fortemente dependente do depurador**
  - Como os erros podem estar em qualquer lugar, é preciso usar o depurador para encontrá-los, o que é trabalhoso
  - Muitos erros de uma vez fazem ser mais difícil resolver cada um
    - » *Pode até surgir a necessidade de modificar a arquitetura do programa nas fases finais do desenvolvimento*

Testes de software – Histórico

- **Demorado e fortemente dependente do depurador**
  - Como os erros podem estar em qualquer lugar, é preciso usar o depurador para encontrá-los, o que é trabalhoso
  - Muitos erros de uma vez fazem ser mais difícil resolver cada um
    - » *Pode até surgir a necessidade de modificar a arquitetura do programa nas fases finais do desenvolvimento*
- **No fim das contas, muitos erros "escapavam" (ou não eram corrigidos por serem considerados "não-críticos")**

A partir de meados dos anos 2000, adoção crescente de testes automatizados por empresas e especialistas em técnicas de gestão do desenvolvimento

Testes de software – Histórico

Um bom processo hoje envolve

Testes de software – Histórico

Um bom processo hoje envolve

- **Escrita de um pedaço pequeno do programa (por exemplo, uma função)**

Testes de software – Histórico

Um bom processo hoje envolve

- **Escrita de um pedaço pequeno do programa (por exemplo, uma função)**
- **Escrita de testes automatizados para esse pedaço**

Testes de software – Histórico

Um bom processo hoje envolve

- **Escrita de um pedaço pequeno do programa (por exemplo, uma função)**
- **Escrita de testes automatizados para esse pedaço**
  - Alguns desenvolvedores preferem implementar os testes antes da função!

Testes de software – Histórico

Um bom processo hoje envolve

- **Escrita de um pedaço pequeno do programa (por exemplo, uma função)**
- **Escrita de testes automatizados para esse pedaço**
  - Alguns desenvolvedores preferem implementar os testes antes da função!
- **Os testes são executados com frequência (no mínimo uma vez por dia, mas em geral várias vezes no mesmo dia)**

Testes de software – Histórico

Um bom processo hoje envolve

- **Escrita de um pedaço pequeno do programa (por exemplo, uma função)**
- **Escrita de testes automatizados para esse pedaço**
  - Alguns desenvolvedores preferem implementar os testes antes da função!
- **Os testes são executados com frequência (no mínimo uma vez por dia, mas em geral várias vezes no mesmo dia)**
- **Os principais erros aparecem "imediatamente"**

Testes de software – Histórico

Um bom processo hoje envolve

- **Escrita de um pedaço pequeno do programa (por exemplo, uma função)**
- **Escrita de testes automatizados para esse pedaço**
  - Alguns desenvolvedores preferem implementar os testes antes da função!
- **Os testes são executados com frequência (no mínimo uma vez por dia, mas em geral várias vezes no mesmo dia)**
- **Os principais erros aparecem "imediatamente"**
  - Se forem erros na sua função, seus testes vão apontar o problema

Testes de software – Histórico

Um bom processo hoje envolve

- **Escrita de um pedaço pequeno do programa (por exemplo, uma função)**
- **Escrita de testes automatizados para esse pedaço**
  - Alguns desenvolvedores preferem implementar os testes antes da função!
- **Os testes são executados com frequência (no mínimo uma vez por dia, mas em geral várias vezes no mesmo dia)**
- **Os principais erros aparecem "imediatamente"**
  - Se forem erros na sua função, seus testes vão apontar o problema
  - Se forem erros na integração da sua função com o restante do programa, algum *outro* teste já existente vai apontar o problema

Testes de software – Histórico

- **Os erros são consertados antes de o novo pedaço do programa ser incorporado definitivamente**

Testes de software – Histórico

- **Os erros são consertados antes de o novo pedaço do programa ser incorporado definitivamente**
  - No mesmo momento em que o pedaço está sendo criado $\longrightarrow$ “está tudo na cabeça”

Testes de software – Histórico

- **Os erros são consertados antes de o novo pedaço do programa ser incorporado definitivamente**
  - No mesmo momento em que o pedaço está sendo criado $\longrightarrow$ “está tudo na cabeça”
- **Usa-se muito menos o depurador**

Testes de software – Histórico

- **Os erros são consertados antes de o novo pedaço do programa ser incorporado definitivamente**
  - No mesmo momento em que o pedaço está sendo criado $\longrightarrow$ “está tudo na cabeça”
- **Usa-se muito menos o depurador**
- **O progresso é mais rápido**

Testes de software – Histórico

- **Os erros são consertados antes de o novo pedaço do programa ser incorporado definitivamente**
  - No mesmo momento em que o pedaço está sendo criado $\longrightarrow$ “está tudo na cabeça”
- **Usa-se muito menos o depurador**
- **O progresso é mais rápido**
- **O resultado final tem menos bugs**

Testes de software

- **Testes automatizados**

Testes de software

- **Testes automatizados**
  - Resolvem os problemas dos testes manuais

Testes de software

- **Testes automatizados**
  - Resolvem os problemas dos testes manuais
    - » *Por serem formalizados, o programador naturalmente vai abordar mais casos*

Testes de software

- **Testes automatizados**
  - Resolvem os problemas dos testes manuais
    - » *Por serem formalizados, o programador naturalmente vai abordar mais casos*
    - » *Podem ser executados frequentemente (várias vezes ao dia)*

Testes de software

- **Testes automatizados**
  - Resolvem os problemas dos testes manuais
    - » *Por serem formalizados, o programador naturalmente vai abordar mais casos*
    - » *Podem ser executados frequentemente (várias vezes ao dia)*
  - Ajudam no próprio desenvolvimento a pensar em todos os casos que podem causar problemas

Testes de software

- **Testes automatizados**
  - Resolvem os problemas dos testes manuais
    - » *Por serem formalizados, o programador naturalmente vai abordar mais casos*
    - » *Podem ser executados frequentemente (várias vezes ao dia)*
  - Ajudam no próprio desenvolvimento a pensar em todos os casos que podem causar problemas
    - » *E se o usuário digitar um dado inválido?*

Testes de software

- **Testes automatizados**
  - Resolvem os problemas dos testes manuais
    - » *Por serem formalizados, o programador naturalmente vai abordar mais casos*
    - » *Podem ser executados frequentemente (várias vezes ao dia)*
  - Ajudam no próprio desenvolvimento a pensar em todos os casos que podem causar problemas
    - » *E se o usuário digitar um dado inválido?*
    - » *E se a lista só tiver um elemento? Ou nenhum?*

Testes de software

- **Testes automatizados**
  - Resolvem os problemas dos testes manuais
    - » *Por serem formalizados, o programador naturalmente vai abordar mais casos*
    - » *Podem ser executados frequentemente (várias vezes ao dia)*
  - Ajudam no próprio desenvolvimento a pensar em todos os casos que podem causar problemas
    - » *E se o usuário digitar um dado inválido?*
    - » *E se a lista só tiver um elemento? Ou nenhum?*
    - » *...*

Testes de software

- **Testes automatizados**
  - Resolvem os problemas dos testes manuais
    - » *Por serem formalizados, o programador naturalmente vai abordar mais casos*
    - » *Podem ser executados frequentemente (várias vezes ao dia)*
  - Ajudam no próprio desenvolvimento a pensar em todos os casos que podem causar problemas
    - » *E se o usuário digitar um dado inválido?*
    - » *E se a lista só tiver um elemento? Ou nenhum?*
    - » *...*
  - Ajudam a organizar o programa de uma maneira melhor

Testes de software

- **Testes automatizados**
  - Resolvem os problemas dos testes manuais
    - » *Por serem formalizados, o programador naturalmente vai abordar mais casos*
    - » *Podem ser executados frequentemente (várias vezes ao dia)*
  - Ajudam no próprio desenvolvimento a pensar em todos os casos que podem causar problemas
    - » *E se o usuário digitar um dado inválido?*
    - » *E se a lista só tiver um elemento? Ou nenhum?*
    - » *...*
  - Ajudam a organizar o programa de uma maneira melhor
    - » *Um programa em que as partes estão divididas de maneira ruim é mais difícil de testar*

Testes de software

- **Testes automatizados**
  - Resolvem os problemas dos testes manuais
    - » *Por serem formalizados, o programador naturalmente vai abordar mais casos*
    - » *Podem ser executados frequentemente (várias vezes ao dia)*
  - Ajudam no próprio desenvolvimento a pensar em todos os casos que podem causar problemas
    - » *E se o usuário digitar um dado inválido?*
    - » *E se a lista só tiver um elemento? Ou nenhum?*
    - » *...*
  - Ajudam a organizar o programa de uma maneira melhor
    - » *Um programa em que as partes estão divididas de maneira ruim é mais difícil de testar*
  - Parecem mais trabalhosos inicialmente, mas o esforço é compensado rapidamente!

Arcabouços de testes e o pytest

- Existem diversos **arcabouços** para o desenvolvimento de testes

Arcabouços de testes e o pytest

- **Existem diversos arcabouços para o desenvolvimento de testes**
- **Em python, o mais comum é o pytest**
  - docs.pytest.org

Arcabouços de testes e o pytest

- **Existem diversos arcabouços para o desenvolvimento de testes**
- **Em python, o mais comum é o pytest**
  - docs.pytest.org
- **Com pytest, você escreve seu programa e também funções chamadas `test_*` ou `*_test` responsáveis por executar os testes desejados**

Arcabouços de testes e o pytest

- **Existem diversos arcabouços para o desenvolvimento de testes**
- **Em python, o mais comum é o pytest**
  - docs.pytest.org
- **Com pytest, você escreve seu programa e também funções chamadas `test_*` ou `*_test` responsáveis por executar os testes desejados**
- **Ao rodar `pytest meuprograma.py`, essas funções são executadas e os resultados são apresentados**

pytest — recursos essenciais

pytest — recursos essenciais

```
import pytest
```

pytest — recursos essenciais

```python
import pytest

def mais1(x):
    return x +1
```

pytest — recursos essenciais

```python
import pytest

def mais1(x):
    return x +1

def test_mais1():
    assert mais1(3) == 4
```

pytest — recursos essenciais

```python
import pytest

def mais1(x):
    return x +1

def test_mais1():
    assert mais1(3) == 4
```

```
$ pytest meuarquivo.py
```

pytest — recursos essenciais

```python
import pytest

def mais1(x):
    return x +1

def test_mais1():
    assert mais1(3) == 4
```

```
$ pytest meuarquivo.py
======================== test session starts ========================
platform linux -- Python 3.11.2, pytest-7.2.1, pluggy-1.0.0+repack
rootdir: /tmp
plugins: anyio-3.6.2
collected 1 item

meuarquivo.py .                                                [100%]

========================= 1 passed in 0.00s =========================
```

pytest — recursos essenciais

```python
import pytest

def mais1(x):
    return x +1

def test_mais1():
    assert mais1(3) == 4
```

```
$ pytest meuarquivo.py
======================== test session starts ========================
platform linux -- Python 3.11.2, pytest-7.2.1, pluggy-1.0.0+repack
rootdir: /tmp
plugins: anyio-3.6.2
collected 1 item

meuarquivo.py .                                                [100%]

========================= 1 passed in 0.00s =========================
```

pytest — recursos essenciais

```python
import pytest

def mais1(x):
    return x *1

def test_mais1():
    assert mais1(3) == 4
```

pytest — recursos essenciais

```
$ pytest meuarquivo.py
```

pytest — recursos essenciais

```
$ pytest meuarquivo.py
======================== test session starts ========================
platform linux -- Python 3.11.2, pytest-7.2.1, pluggy-1.0.0+repack
rootdir: /tmp
plugins: anyio-3.6.2
collected 1 item

meuarquivo.py F                                              [100%]
```

pytest — recursos essenciais

```
$ pytest meuarquivo.py
======================== test session starts ========================
platform linux -- Python 3.11.2, pytest-7.2.1, pluggy-1.0.0+repack
rootdir: /tmp
plugins: anyio-3.6.2
collected 1 item

meuarquivo.py F                                              [100%]
```

pytest — recursos essenciais

```
$ pytest meuarquivo.py
======================== test session starts ========================
platform linux -- Python 3.11.2, pytest-7.2.1, pluggy-1.0.0+repack
rootdir: /tmp
plugins: anyio-3.6.2
collected 1 item

meuarquivo.py F                                                [100%]

============================= FAILURES ==============================
_____________________________ test_mais1 _____________________________

    def test_mais1():
>       assert mais1(3) == 4
E       assert 3 == 4
E        +  where 3 = mais1(3)

meuarquivo.py:4: AssertionError
====================== short test summary info ======================
FAILED meuarquivo.py::test_mais1 - assert 3 == 4
======================== 1 failed in 0.00s ==========================
```

pytest — recursos essenciais

- Esse teste verifica a adição para um inteiro só

pytest — recursos essenciais

- **Esse teste verifica a adição para um inteiro só**
  - Mas não seria bom testar também negativos? Zero? floats?

pytest — recursos essenciais

- **Esse teste verifica a adição para um inteiro só**
  - Mas não seria bom testar também negativos? Zero? floats?
- **É possível criar mais testes com esses outros valores, ou...**

pytest — recursos essenciais

- **Esse teste verifica a adição para um inteiro só**
  - Mas não seria bom testar também negativos? Zero? floats?
- **É possível criar mais testes com esses outros valores, ou...**

```python
import pytest

def mais1(x):
    return x +1
```

pytest — recursos essenciais

- **Esse teste verifica a adição para um inteiro só**
  - Mas não seria bom testar também negativos? Zero? floats?
- **É possível criar mais testes com esses outros valores, ou...**

```python
import pytest

def mais1(x):
    return x +1

@pytest.mark.parametrize("entrada, saída", [(2, 3), (0, 1), (-1, 0),
                                            (-2, -1), (1.5, 2.5)])
```

pytest — recursos essenciais

- **Esse teste verifica a adição para um inteiro só**
  - Mas não seria bom testar também negativos? Zero? floats?
- **É possível criar mais testes com esses outros valores, ou...**

```python
import pytest

def mais1(x):
    return x +1

@pytest.mark.parametrize("entrada, saída", [(2, 3), (0, 1), (-1, 0),
                                            (-2, -1), (1.5, 2.5)])
def test_mais1(entrada, saída):
    assert mais1(entrada) == saída
```

pytest — recursos essenciais

- Podemos ter zero, um, dois ou mais parâmetros

pytest — recursos essenciais

- Podemos ter zero, um, dois ou mais parâmetros

```
import pytest

def medias(x, y):
    return (x + y) /2, (x * y)**0.5
```

pytest — recursos essenciais

- Podemos ter zero, um, dois ou mais parâmetros

```
import pytest

def medias(x, y):
    return (x + y) /2, (x * y)**0.5

@pytest.mark.parametrize("entrada1, entrada2, saída1, saída2",
                         [(-1, 0, -0.5, 0), (2, 4, 3, 2.82843), (0, 1, 0.5, 0)
                          (1.5, 2.5, 2, 1.93649), (-2, -1, -1.5, 1.41421)])
```

pytest — recursos essenciais

- Podemos ter zero, um, dois ou mais parâmetros

```python
import pytest

def medias(x, y):
    return (x + y) /2, (x * y)**0.5

@pytest.mark.parametrize("entrada1, entrada2, saída1, saída2",
                         [(-1, 0, -0.5, 0), (2, 4, 3, 2.82843), (0, 1, 0.5, 0)
                          (1.5, 2.5, 2, 1.93649), (-2, -1, -1.5, 1.41421)])
def test_medias(entrada1, entrada2, saída1, saída2):
    a, b = medias(entrada1, entrada2)
    assert round(a, 5) == round(saída1, 5)
    assert round(b, 5) == round(saída2, 5)
```

pytest — recursos essenciais

- Com isso, podemos testar funções que recebem parâmetros e devolvem valores

pytest — recursos essenciais

- Com isso, podemos testar funções que recebem parâmetros e devolvem valores
- Mas como testar funções que usam `print()`?

pytest — recursos essenciais

- Com isso, podemos testar funções que recebem parâmetros e devolvem valores
- Mas como testar funções que usam `print()`?

```
import pytest

def exclama(s):
    print(str(s) + "!")
```

pytest — recursos essenciais

- Com isso, podemos testar funções que recebem parâmetros e devolvem valores
- Mas como testar funções que usam `print()`?

```
import pytest

def exclama(s):
    print(str(s) + "!")

@pytest.mark.parametrize("entrada, saída", [("oi", "oi!\n"), (5, "5!\n")])
```

pytest — recursos essenciais

- Com isso, podemos testar funções que recebem parâmetros e devolvem valores
- Mas como testar funções que usam `print()`?

```python
import pytest

def exclama(s):
    print(str(s) + "!")

@pytest.mark.parametrize("entrada, saída", [("oi", "oi!\n"), (5, "5!\n")])
def test_exclama(capsys, entrada, saída):
```

pytest — recursos essenciais

- Com isso, podemos testar funções que recebem parâmetros e devolvem valores
- Mas como testar funções que usam `print()`?

```python
import pytest

def exclama(s):
    print(str(s) + "!")

@pytest.mark.parametrize("entrada, saída", [("oi", "oi!\n"), (5, "5!\n")])
def test_exclama(capsys, entrada, saída):
    exclama(entrada)
```

pytest — recursos essenciais

- Com isso, podemos testar funções que recebem parâmetros e devolvem valores
- Mas como testar funções que usam `print()`?

```python
import pytest

def exclama(s):
    print(str(s) + "!")

@pytest.mark.parametrize("entrada, saída", [("oi", "oi!\n"), (5, "5!\n")])
def test_exclama(capsys, entrada, saída):
    exclama(entrada)
    out, err = capsys.readouterr()
    assert out == saída
```

pytest — recursos essenciais

- Só faltam funções que usam `input()`...

pytest — recursos essenciais

- Só faltam funções que usam `input()`...

```
import pytest
import io
```

pytest — recursos essenciais

- Só faltam funções que usam `input()`...

```
import pytest
import io

def amor():
    print(input(), "ama", input())
```

pytest — recursos essenciais

- Só faltam funções que usam `input()`...

```
import pytest
import io

def amor():
    print(input(), "ama", input())

@pytest.mark.parametrize("fulano, ciclano, saída", [("John", "Yoko", "John ama Yoko"),
                         ("Tristão", "Isolda", "Tristão ama Isolda"),
                         ("Frida", "Josephine", "Frida ama Josephine")])
```

pytest — recursos essenciais

- Só faltam funções que usam `input()`...

```
import pytest
import io

def amor():
    print(input(), "ama", input())

@pytest.mark.parametrize("fulano, ciclano, saída", [("John", "Yoko", "John ama Yoko"),
                          ("Tristão", "Isolda", "Tristão ama Isolda"),
                          ("Frida", "Josephine", "Frida ama Josephine")])
def test_amor(monkeypatch, capsys, fulano, ciclano, saída):
```

pytest — recursos essenciais

- Só faltam funções que usam `input()`...

```python
import pytest
import io

def amor():
    print(input(), "ama", input())

@pytest.mark.parametrize("fulano, ciclano, saída", [("John", "Yoko", "John ama Yoko"),
                         ("Tristão", "Isolda", "Tristão ama Isolda"),
                         ("Frida", "Josephine", "Frida ama Josephine")])
def test_amor(monkeypatch, capsys, fulano, ciclano, saída):
    nomes = io.StringIO(str(fulano) + "\n" + str(ciclano) + "\n")
    monkeypatch.setattr('sys.stdin', nomes)
```

pytest — recursos essenciais

- Só faltam funções que usam `input()`...

```python
import pytest
import io

def amor():
    print(input(), "ama", input())

@pytest.mark.parametrize("fulano, ciclano, saída", [("John", "Yoko", "John ama Yoko"),
                          ("Tristão", "Isolda", "Tristão ama Isolda"),
                          ("Frida", "Josephine", "Frida ama Josephine")])
def test_amor(monkeypatch, capsys, fulano, ciclano, saída):
    nomes = io.StringIO(str(fulano) + "\n" + str(ciclano) + "\n")
    monkeypatch.setattr('sys.stdin', nomes)
    amor()
```

pytest — recursos essenciais

- Só faltam funções que usam `input()`...

```python
import pytest
import io

def amor():
    print(input(), "ama", input())

@pytest.mark.parametrize("fulano, ciclano, saída", [("John", "Yoko", "John ama Yoko"),
                         ("Tristão", "Isolda", "Tristão ama Isolda"),
                         ("Frida", "Josephine", "Frida ama Josephine")])
def test_amor(monkeypatch, capsys, fulano, ciclano, saída):
    nomes = io.StringIO(str(fulano) + "\n" + str(ciclano) + "\n")
    monkeypatch.setattr('sys.stdin', nomes)
    amor()
    out, err = capsys.readouterr()
    assert out == saída + "\n"
```

pytest — recursos essenciais

- **Em geral, é melhor colocar os testes em um arquivo separado**

meuarquivo.py

```python
def mais1(x):
    return x +1

def exclama(s):
    print(str(s) + "!")

def amor():
    print(input(), "ama ", input())

def main():
    exclama(mais1(5))

if __name__ == "__main__":
    main()
```

pytest — recursos essenciais

- **Em geral, é melhor colocar os testes em um arquivo separado**

meuarquivo.py

```python
def mais1(x):
    return x +1

def exclama(s):
    print(str(s) + "!")

def amor():
    print(input(),"ama ", input())

def main():
    exclama(mais1(5))

if __name__ == "__main__":
    main()
```

pytest — recursos essenciais

- **Em geral, é melhor colocar os testes em um arquivo separado**

test_meuarquivo.py

```
import pytest
import meuarquivo
@pytest.mark.parametrize("entrada, saída", [(2, 3), (-2, -1), (0, 1), (-1, 0), (1.5, 2.5)])
def test_mais1(entrada, saída):
    assert saída == meuarquivo.mais1(entrada)
@pytest.mark.parametrize("entrada, saída", [("oi", "oi!\n"), (5, "5!\n")])
def test_exclama(capsys, entrada, saída):
    meuarquivo.exclama(entrada)
    resultado = capsys.readouterr().out
    assert resultado == saída
@pytest.mark.parametrize("fulano, ciclano, saída", [("John", "Yoko", "John ama Yoko"),
    ("Tristão", "Isolda", "Tristão ama Isolda"), ("Frida", "Josephine", "Frida ama Josephine")])
def test_amor(monkeypatch, capsys, fulano, ciclano, saída):
    nomes = io.StringIO(str(fulano) + "\n" + str(ciclano) + "\n")
    monkeypatch.setattr('sys.stdin', nomes)
    meuarquivo.amor()
    out, err = capsys.readouterr()
    assert out == saída + "\n"
```

pytest — recursos essenciais

- Sem parâmetros, pytest processa automaticamente arquivos de nome `test_*.py` ou `*_test.py`

pytest — recursos essenciais

- Sem parâmetros, pytest processa automaticamente arquivos de nome `test_*.py` ou `*_test.py`

```
$ pytest
```

pytest — recursos essenciais

- **Sem parâmetros, pytest processa automaticamente arquivos de nome `test_*.py` ou `*_test.py`**

```
$ pytest
======================= test session starts ========================
platform linux -- Python 3.11.2, pytest-7.2.1, pluggy-1.0.0+repack
rootdir: /tmp
plugins: anyio-3.6.2
collected 10 items

test_meuarquivo.py .......FFF                                  [100%]

============================= FAILURES =============================
…
====================== short test summary info ======================
FAILED test_meuarquivo.py::test_amor[John-Yoko-John ama Yoko] - AssertionError: assert 'John ama  Yoko\n' ==
'John ama Yoko\n'
FAILED test_meuarquivo.py::test_amor[Trist\xe3o-Isolda-Trist\xe3o ama Isolda] - AssertionError: assert
'Tristão ama  Isolda\n' == 'Tristão ama Isolda\n'
FAILED test_meuarquivo.py::test_amor[Frida-Josephine-Frida ama Josephine] - AssertionError: assert 'Frida ama
Josephine\n' == 'Frida ama Josephine\n'
==================== 3 failed, 7 passed in 0.00s ====================
```

pytest — recursos essenciais

- **Sem parâmetros, pytest processa automaticamente arquivos de nome `test_*.py` ou `*_test.py`**

```
$ pytest
======================= test session starts ========================
platform linux -- Python 3.11.2, pytest-7.2.1, pluggy-1.0.0+repack
rootdir: /tmp
plugins: anyio-3.6.2
collected 10 items

test_meuarquivo.py .......FFF                                    [100%]

============================= FAILURES =============================
…
====================== short test summary info ======================
FAILED test_meuarquivo.py::test_amor[John-Yoko-John ama Yoko] - AssertionError: assert 'John ama  Yoko\n' ==
'John ama Yoko\n'
FAILED test_meuarquivo.py::test_amor[Trist\xe3o-Isolda-Trist\xe3o ama Isolda] - AssertionError: assert
'Tristão ama  Isolda\n' == 'Tristão ama Isolda\n'
FAILED test_meuarquivo.py::test_amor[Frida-Josephine-Frida ama Josephine] - AssertionError: assert 'Frida ama
Josephine\n' == 'Frida ama Josephine\n'
==================== 3 failed, 7 passed in 0.00s ====================
```

O que testar?

O que testar?

O que testar?

O que testar?

- **Amostras de todos os casos comuns; por exemplo:**

O que testar?

O que testar?

- **Amostras de todos os casos comuns; por exemplo:**
  - Inteiros positivos e negativos

O que testar?

O que testar?

- **Amostras de todos os casos comuns; por exemplo:**
  - Inteiros positivos e negativos
    - » *Com um único dígito*

O que testar?

O que testar?

- **Amostras de todos os casos comuns; por exemplo:**
  - Inteiros positivos e negativos
    - » *Com um único dígito*
    - » *Com dois dígitos*

O que testar?

O que testar?

- **Amostras de todos os casos comuns; por exemplo:**
  - Inteiros positivos e negativos
    - » *Com um único dígito*
    - » *Com dois dígitos*
    - » *Com vários dígitos (p. ex., 5)*

O que testar?

O que testar?

- **Amostras de todos os casos comuns; por exemplo:**
  - Inteiros positivos e negativos
    - » *Com um único dígito*
    - » *Com dois dígitos*
    - » *Com vários dígitos (p. ex., 5)*
    - » *Com dígitos repetidos*

O que testar?

O que testar?

- **Amostras de todos os casos comuns; por exemplo:**
  - Inteiros positivos e negativos
    - » *Com um único dígito*
    - » *Com dois dígitos*
    - » *Com vários dígitos (p. ex., 5)*
    - » *Com dígitos repetidos*
    - » *Múltiplos e não-múltiplos de 10 e de 2*

O que testar?

O que testar?

- **Amostras de todos os casos comuns; por exemplo:**
  - Inteiros positivos e negativos
    - » *Com um único dígito*
    - » *Com dois dígitos*
    - » *Com vários dígitos (p. ex., 5)*
    - » *Com dígitos repetidos*
    - » *Múltiplos e não-múltiplos de 10 e de 2*
    - » *Primos e não-primos*

O que testar?

O que testar?

- **Amostras de todos os casos comuns; por exemplo:**
  - Inteiros positivos e negativos
    - » *Com um único dígito*
    - » *Com dois dígitos*
    - » *Com vários dígitos (p. ex., 5)*
    - » *Com dígitos repetidos*
    - » *Múltiplos e não-múltiplos de 10 e de 2*
    - » *Primos e não-primos*
  - Floats positivos e negativos

O que testar?

O que testar?

- **Amostras de todos os casos comuns; por exemplo:**
  - Inteiros positivos e negativos
    - » *Com um único dígito*
    - » *Com dois dígitos*
    - » *Com vários dígitos (p. ex., 5)*
    - » *Com dígitos repetidos*
    - » *Múltiplos e não-múltiplos de 10 e de 2*
    - » *Primos e não-primos*
  - Floats positivos e negativos
    - » *"Grandes" e "pequenos"*

O que testar?

O que testar?

- **Amostras de todos os casos comuns; por exemplo:**
  - Inteiros positivos e negativos
    - » *Com um único dígito*
    - » *Com dois dígitos*
    - » *Com vários dígitos (p. ex., 5)*
    - » *Com dígitos repetidos*
    - » *Múltiplos e não-múltiplos de 10 e de 2*
    - » *Primos e não-primos*
  - Floats positivos e negativos
    - » *"Grandes" e "pequenos"*
    - » *Racionais exatos, como 0,25*

O que testar?

O que testar?

- **Amostras de todos os casos comuns; por exemplo:**
  - Inteiros positivos e negativos
    - » *Com um único dígito*
    - » *Com dois dígitos*
    - » *Com vários dígitos (p. ex., 5)*
    - » *Com dígitos repetidos*
    - » *Múltiplos e não-múltiplos de 10 e de 2*
    - » *Primos e não-primos*
  - Floats positivos e negativos
    - » *"Grandes" e "pequenos"*
    - » *Racionais exatos, como 0,25*
    - » *Irracionais ou racionais aproximados, como 0,1 ou* $\sqrt{2}$

O que testar?

O que testar?

O que testar?

O que testar?

- **Quantidades “especiais”; por exemplo:**

O que testar?

O que testar?

- **Quantidades “especiais”; por exemplo:**
  - Números zero, 1 e -1

O que testar?

O que testar?

- **Quantidades “especiais”; por exemplo:**
  - Números zero, 1 e -1
  - Listas/strings com zero ou um elementos/caracteres

O que testar?

O que testar?

- **Quantidades “especiais”; por exemplo:**
  - Números zero, 1 e -1
  - Listas/strings com zero ou um elementos/caracteres
  - As coordenadas nas “pontas” de uma matriz

O que testar?

O que testar?

- **Quantidades “especiais”; por exemplo:**
  - Números zero, 1 e -1
  - Listas/strings com zero ou um elementos/caracteres
  - As coordenadas nas “pontas” de uma matriz
- **Casos que seu programa trata de maneira especial; por exemplo:**

O que testar?

O que testar?

- **Quantidades “especiais”; por exemplo:**
  - Números zero, 1 e -1
  - Listas/strings com zero ou um elementos/caracteres
  - As coordenadas nas “pontas” de uma matriz
- **Casos que seu programa trata de maneira especial; por exemplo:**
  - `éPrimo(x)` trata o número 2 de maneira especial

O que testar?

O que testar?

- **Quantidades “especiais”; por exemplo:**
  - Números zero, 1 e -1
  - Listas/strings com zero ou um elementos/caracteres
  - As coordenadas nas “pontas” de uma matriz
- **Casos que seu programa trata de maneira especial; por exemplo:**
  - `éPrimo(x)` trata o número 2 de maneira especial
- **Casos de erro ou dados inválidos; por exemplo:**

O que testar?

O que testar?

- **Quantidades “especiais”; por exemplo:**
  - Números zero, 1 e -1
  - Listas/strings com zero ou um elementos/caracteres
  - As coordenadas nas “pontas” de uma matriz
- **Casos que seu programa trata de maneira especial; por exemplo:**
  - `éPrimo(x)` trata o número 2 de maneira especial
- **Casos de erro ou dados inválidos; por exemplo:**
  - Usuário informou idade negativa

O que testar?

O que testar?

- **Quantidades “especiais”; por exemplo:**
  - Números zero, 1 e -1
  - Listas/strings com zero ou um elementos/caracteres
  - As coordenadas nas “pontas” de uma matriz
- **Casos que seu programa trata de maneira especial; por exemplo:**
  - `éPrimo(x)` trata o número 2 de maneira especial
- **Casos de erro ou dados inválidos; por exemplo:**
  - Usuário informou idade negativa
  - A função `calcula_raiz_segundo_grau()` recebeu uma equação sem raízes reais

O que testar?

O que testar?

- **Quantidades “especiais”; por exemplo:**
  - Números zero, 1 e -1
  - Listas/strings com zero ou um elementos/caracteres
  - As coordenadas nas “pontas” de uma matriz
- **Casos que seu programa trata de maneira especial; por exemplo:**
  - `éPrimo(x)` trata o número 2 de maneira especial
- **Casos de erro ou dados inválidos; por exemplo:**
  - Usuário informou idade negativa
  - A função `calcula_raiz_segundo_grau()` recebeu uma equação sem raízes reais
  - Um arquivo necessário não existe

Testes de software — exercícios

Quais testes você criaria para

Testes de software — exercícios

Quais testes você criaria para

- **A função `fatorial()`?**

Testes de software — exercícios

Quais testes você criaria para

- A função `fatorial()`?
- A função `éPrimo()`?

Testes de software — exercícios

Quais testes você criaria para

- **A função `fatorial()`?**
- **A função `éPrimo()`?**
- **Uma função que recebe uma lista e um valor e devolve o índice da primeira ocorrência do valor na lista ou -1 caso o valor não esteja presente na lista?**