MAC 110 — Introdução à Ciência da Computação

Aula 8

Nelson Lago

BMAC – 2024

Exercícios

```
n = 42
a = 0
while n >= a:
    b = n // 2
    if b * 2 == n:
        a = a + 3
    else:
        a = a + 2
    n = n - 7
    x = a % 9
    print(x * 7 + 13)
```

S1:	☐	N/A	☐	48	☐	41	☐	62	☐	55	☐	34
S2:	☐	55	☐	N/A	☐	62	☐	69	☐	48	☐	13
S3:	☐	20	☐	27	☐	13	☐	N/A	☐	69	☐	34
S4:	☐	27	☐	41	☐	34	☐	N/A	☐	20	☐	48
S5:	☐	41	☐	48	☐	55	☐	69	☐	N/A	☐	62

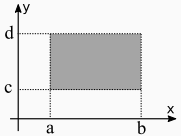
Dadas duas variáveis inteiras a e b, selecione todas as expressões equivalentes a: `a >= b`

- [ ] `a < b == False`
- [ ] `a > b + 1`
- [ ] `a > b and a == b`
- [ ] `a > b - 1`
- [ ] `a > b or a == b`
- [ ] `(a // 2) >= (b // 2)`
- [ ] `not (a <= b or a != b)`
- [ ] `not (a <= b and a != b)`
- [ ] `not (a < b)`

Dada a nota de um aluno em uma variável real n, selecione todas as expressões equivalentes a: `n < 3.0 or n >= 5.0`

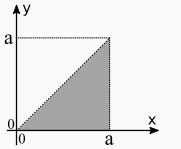
- [ ] `(n - 4.0)**2 > 1.0 or n == 5.0`
- [ ] `not (n >= 3.0 and n < 5.0)`
- [ ] `n - 4.0 >= 1.0 and n - 4.0 < -1.0`
- [ ] `(n - 4.0)**2 > 1.0 and n - 4.0 != 1.0`
- [ ] `not (n >= 3.0 or n < 5.0)`
- [ ] `n - 4.0 >= 1.0 or n - 4.0 < -1.0`
- [ ] `n >= 3.0 and n < 5.0`
- [ ] `n < 3.0 or n > 5.0 or n == 5.0`

Dadas as coordenadas reais x e y de um ponto, selecione todas expressões que geram `True` se esse ponto está na região sombreada da gura ao lado e `False` caso contrário. A região sombreada não inclui as linhas de fronteira.



- [ ] `x > a and x < b and y > c and y < d`
- [ ] `(x<b and y<d) or not (x<=a and y<=c)`
- [ ] `(x<b and y<d) and not (x<=a or y<=c)`
- [ ] `not (x<=a and x>=b and y<=c and y>=d)`
- [ ] `x > a or x < b or y > c or y < d`
- [ ] `not (x<=a and x>=b) or not (y<=c and y>=d)`
- [ ] `not (x<=a or x>=b) and not (y<=c or y>=d)`
- [ ] `not (x<=a or x>=b or y<=c or y>=d)`

Dadas as coordenadas reais x e y de um ponto, selecione todas expressões que geram `True` se esse ponto está na região sombreada da gura ao lado e `False` caso contrário. A região sombreada não inclui as linhas de fronteira.



- [ ] `x < a or (0 < y and y < x)`
- [ ] `not (x >= a and y <= 0 and x <= y)`
- [ ] `x >= a or y <= 0 or x <= y`
- [ ] `x < a and y > 0 and not(x <= y)`
- [ ] `y < x and not (x >= a or 0 >= y)`
- [ ] `not (x >= a or y <= 0 or x <= y)`
- [ ] `x < a and y > 0 and x > y`
- [ ] `x > 0 or x < a or y < a or y > 0 or x > y`
- [ ] `x < a or y > 0 or x > y`
- [ ] `x>0 and x<a and y<a and y>0 and x>y`

Dígitos do CPF

O CPF tem a configuração XXX.XXX.XXX-XX, na qual os primeiros nove dígitos (da esquerda para direita) são o número-base e os dois últimos são dígitos de verificação (DV), usados para prevenir erros de transmissão ou digitação. O nono dígito da base define a Região Fiscal onde foi emitido o CPF, tendo a seguinte abrangência: 1 (DF-GO-MS-MT-TO), 2 (AC-AM-AP-PA-RO-RR), 3 (CE-MA-PI), 4 (AL-PB-PE-RN), 5 (BA-SE), 6 (MG), 7 (ES-RJ), 8 (SP), 9 (PR-SC) e 0 (RS). O primeiro DV corresponde ao resto da divisão por 11 do somatório dos algarismos da base, visitados da esquerda para a direita, multiplicados respectivamente pelos pesos 1, 2, 3, 4, 5, 6, 7, 8 e 9. O resto 10 é considerado 0. O segundo DV é calculado de modo análogo, porém acrescentando um dígito extra à direita na base, dado pelo primeiro DV calculado, e iniciando com o peso 0, tal como explicado no exemplo.

Dígitos do CPF

Exemplo: Para CPFs da forma 280.012.389-XX, com base 280012389 e região fiscal 9, temos:

dígito	2	8	0	0	1	2	3	8	9	
peso	1	2	3	4	5	6	7	8	9	
soma	2+	16+	0+	0+	5+	12+	21+	64+	81	= 201

`201 % 11 = 3`, então

dígito	2	8	0	0	1	2	3	8	9	3	
peso	0	1	2	3	4	5	6	7	8	9	
soma	0+	8+	0+	0+	4+	10+	18+	56+	72+	27	= 195

`195 % 11 = 8`, logo o único CPF válido com a base fornecida é 280.012.389-38 (isto é, XX = 38). Portanto, por exemplo, o CPF 280.012.389-14 é inválido.

Dígitos do CPF

Utilizando um único laço while, escolha as alternativas que preenchem as lacunas no código a seguir (L1 até L12), de forma a obter um programa em Python que, dado um número de CPF (sem '-' e '.'), fornecido como um inteiro com 11 dígitos, verifica se ele é um CPF válido pertencente à Região Sul do Brasil (PR-Paraná, SC-Santa Catarina e RS-Rio Grande do Sul).

```
cpf = int(input("Digite o CPF: "))
L1
L2
s1 = s2 = 0
i = 9
while L3:
    L4
    L5
    L6
    L7
    i = i - 1
L8
L9
L10
L11
if L12:
    print("CPF válido da região sul do Brasil")
```

L1:
- ☐ `DVs = cpf%100`
- ☐ `DVs = cpf%10 + cpf%100`
- ☐ `DVs = (cpf*10)%10`
- ☐ `DVs = cpf%10`
- ☐ `DVs = cpf//100`

L2:
- ☐ `base = cpf//100`
- ☐ `base = cpf%10`
- ☐ `base = base - cpf%100`
- ☐ `base = cpf//10`
- ☐ `base = cpf%100`

L3:
- ☐ `cpf >= 0`
- ☐ `base >= 0`
- ☐ `cpf > 0`
- ☐ `i > 1`
- ☐ `base > 0`

L4:
- ☐ `r = base%10`
- ☐ `base = base%10`
- ☐ `r = base//10`
- ☐ `r = cpf%10`
- ☐ `r = cpf//10`
- ☐ `base = base//10`

L5:
- ☐ `s1 = s1 + r*(9-i-1)`
- ☐ `s1 = s1 + r*(9-i)`
- ☐ `s1 = s1 + r*(i+1)`
- ☐ `s1 = s1 + r*(i-1)`
- ☐ `s1 = s1 + r*i`
- ☐ `s1 = s1 + r*(9-i+1)`

L6:
- ☐ `s2 = s2 + r*(i+1)`
- ☐ `s2 = s2 + r*i`
- ☐ `s2 = s2 + r*(9-i-1)`
- ☐ `s2 = s2 + r*(9-i+1)`
- ☐ `s2 = s2 + r*(9-i)`
- ☐ `s2 = s2 + r*(i-1)`

L7:
- ☐ `base = base-r`
- ☐ `base = base-r*10`
- ☐ `base = base%10`
- ☐ `base = base//10`
- ☐ `base = base//(i*10)`

L8:
- ☐ `DV1 = s1%11`
- ☐
```
DV1 = s1
while DV1 > 11:
    DV1 = DV1 - 11
```
- ☐ `DV1 = (s1%11)%10`
- ☐ `DV1 = (s1%11)//10`

L9:
- ☐ `s2 = s2 + DV1*9`
- ☐ `s2 = s2 + r*9`
- ☐ `s2 = s2 + DV1*r`
- ☐ `s2 = s2 + DV1*i`
- ☐ `s2 = s2 + i*9`

L10:
- ☐ `DV2 = (s2%11)//10`
- ☐
```
DV2 = s2
while DV2 > 11:
    DV2 = DV2 - 11
```
- ☐ `DV2 = s2%11`
- ☐ `DV2 = (s2%11)%10`

L11:
- ☐ `reg = cpf//(10**8)`
- ☐ `reg = base%10`
- ☐ `reg = (cpf//100)%10`
- ☐ `reg = cpf - DVs`
- ☐ `reg = base//(10**8)`

L12:
- ☐ `(reg==9 and reg==0) or DVs == DV1*10+DV2`
- ☐ `reg%9 == 0 and DVs == DV1+DV2*10`
- ☐ `(reg==9 or reg==0) and DVs == DV1*10+DV2`
- ☐ `reg==9 or (reg==0 and DVs == DV1*10+DV2)`
- ☐ `reg%9 == 0 and DVs == DV1+DV2`

Trechos de código

Considere os 6 seguintes trechos de código (T1 até T6) e selecione as afirmações verdadeiras sobre eles pintando as quadrículas.

1. **Considere que o usuário SEMPRE digitará uma sequência de naturais entre 1 e 100, terminando com o valor zero (0). Além disso, a sequência (SEQ) tem ao menos 2 elementos.**
2. **As afirmações devem estar corretas quaisquer que sejam as entradas atendendo essas restrições.**
3. **Nos trechos de código da imagem abaixo, o termo SEQ nas afirmações significa "sequência".**
4. **Note que a SEQ {1, 1, 1} deve ser considerada crescente e também decrescente.**
5. **As opções sobre cada trecho podem conter desde nenhuma afirmação correta até todas.**

Trechos de código

```python
# Trecho 1
x = 1
M = 0
while x != 0:
    x = int(input())
    if M < x:
        M = x
print(M)
```

```python
# Trecho 2
x = 1
M = 0
while x != 0:
    if M < x:
        M = x
    x = int(input())
print(M)
```

```python
# Trecho 3
x = 1
M = 0
while x != 0:
    x = int(input())
    if M > x:
        M = x
print(M)
```

T1

- ☐ Seleciona o menor natural em SEQ
- ☐ Seleciona o maior natural em SEQ
- ☐ Determina se a SEQ é crescente
- ☐ Falha ao tentar selecionar o maior natural em SEQ

T2

- ☐ Seleciona o maior natural em SEQ
- ☐ Falha ao tentar selecionar o menor natural em SEQ
- ☐ Seleciona o menor natural em SEQ
- ☐ Pode selecionar o menor para algumas SEQ

T3

- ☐ Falha ao tentar selecionar o menor natural em SEQ
- ☐ Seleciona o menor natural em SEQ

Trechos de código

```python
# Trecho 4
x = 1000
M = 1000
while x != 0:
    if M > x:
        M = x
    x = int(input())
print(M)
```

T4 | ☐ Falha ao tentar selecionar o menor natural em SEQ ☐ Seleciona o menor natural em SEQ

```python
# Trecho 5
x = -1
y = -2
M = 1
while x != 0:
    if y > x:
        M = 0
    y = x
    x = int(input())
if M == 1:
    print("Sim")
else:
    print("Não")
```

```python
# Trecho 6
x = -1
y = -2
M = 1
while x != 0:
    if y > x:
        M = 0
    else:
        M = 1
    y = x
    x = int(input())
if M == 1:
    print("Sim")
else:
    print("Não")
```

T5

- ☐ Imprime sim se a SEQ for estritamente crescente
- ☐ Imprime sim se a SEQ for decrescente
- ☐ Imprime nao se a SEQ for estritamente decrescente
- ☐ Imprime sim se a SEQ for crescente

T6

- ☐ Imprime sim se a SEQ for decrescente
- ☐ Imprime nao se últimos 2 da SEQ são estritamente decrescentes
- ☐ Imprime nao se a SEQ for decrescente
- ☐ Imprime sim se últimos 2 da SEQ são estritamente crescentes