Processamento da Informação

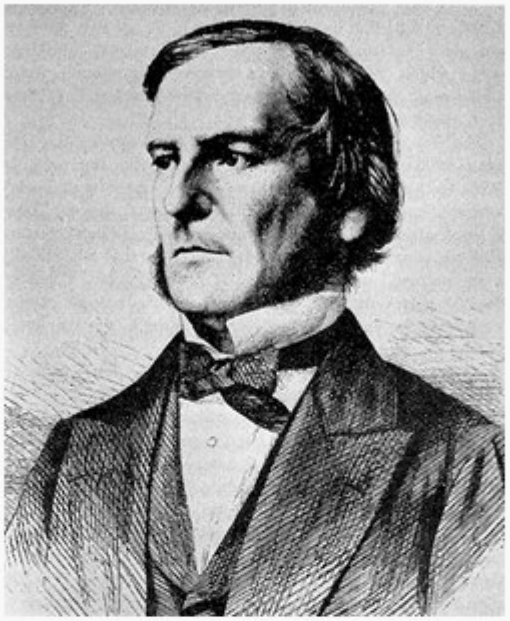
Estruturas de seleção - Parte 2

Prof. Jesús P. Mena-Chalco
CMCC/UFABC

Q1/2020

Expressões Booleanas

George Boole



Nacionalidade	Britânico
Nascimento	2 de Novembro de 1815
Local	Lincoln
Morte	8 de Dezembro de 1864 (49 anos)
Local	Ballintemple
Cônjuge	Mary Everest
Conhecido(a) por	Álgebra booleana
Influenciado(s)	Aristóteles, Spinoza e Newton
Prêmio(s)	Medalha Real (1844)

Fonte: http://pt.wikipedia.org/wiki/George_Boole

Nos 1840, Boole demostrou que as regras clássicas de lógica podem ser expressas de forma puramente matemática usando apenas dois valores **True** e **False**.

A lógica booleada foi muito utilizado para projetar circuitos de computadores.

Tipicamente, existe em toda linguagem de programação moderna um tipo de dado **bool**.

Expressões Booleanas

Uma expressão booleana é uma expressão que é ou **Verdadeira** ou **Falsa**.

Os seguintes exemplos usam o operador "**==**", utilizado para comparar dois operandos e produzir true se eles forem iguais ou false em caso contrário.

- `5 == 5` → `true`
- `5 == 6` → `false`
- `True == True` → `true`

Expressões Booleanas

O operador “==” é um dos operadores relacionais, os outros são:

```
x !=  y       // x não é igual a y
x  >  y       // x é maior que y
x  <  y       // x é menor que y
x >= y        // x é maior ou igual a y
x <= y        // x é menor ou iqual a y
```

Um erro comum é usar “=” no lugar de “==”.

Não existem os operadores **=<** ou **=>**.

Lembrando: Estrutura de seleção

Execução condicional

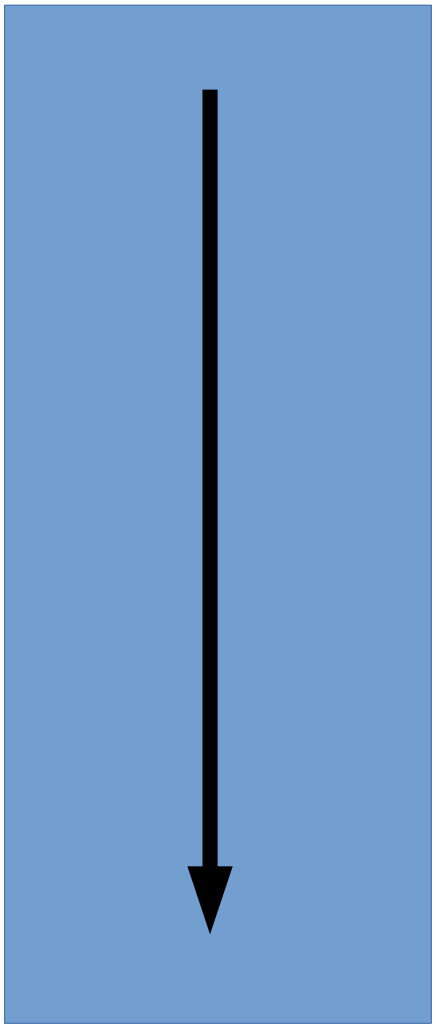
```
Instrucao1
Instrucao2
Instrucao3

if ……… :
    Instrucao-b1
    Instrucao-b2
    Instrucao-b3

Instrucao4
Instrucao5
Instrucao6
Instrucao7
Instrucao8
```

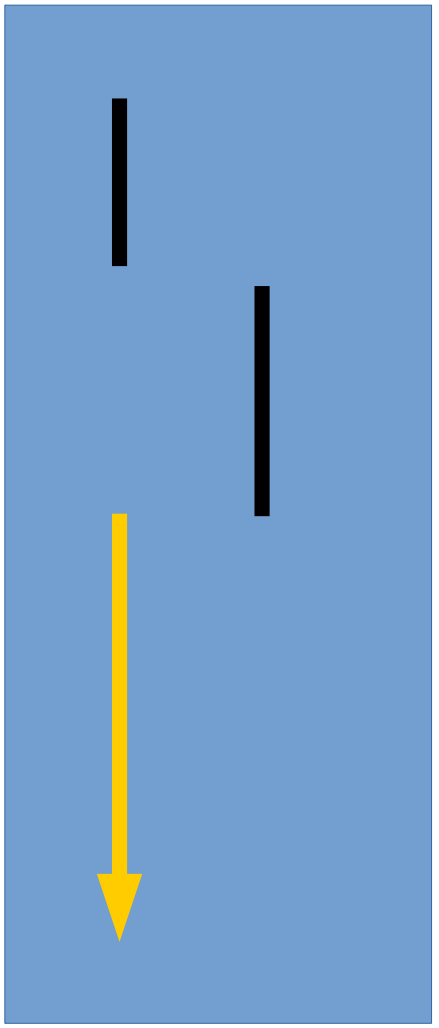
As instruções b1, b2 e b3 serão executadas apenas se a condição for **True**

Execução condicional

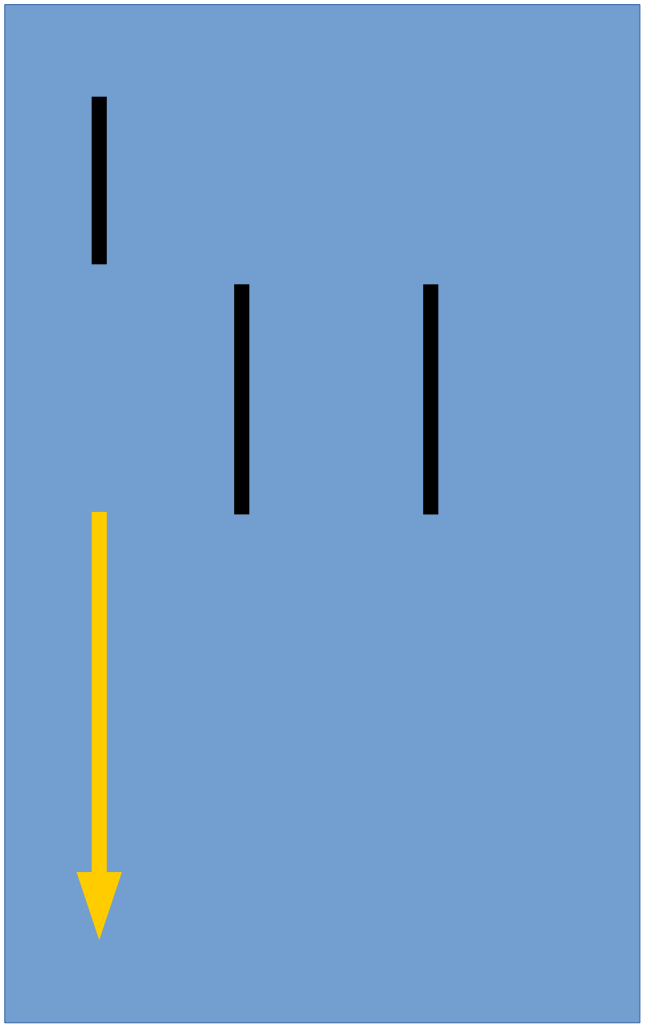
```
Instrucao1
Instrucao2
Instrucao3

if …… :
    Instrucao-b1
    Instrucao-b2
    Instrucao-b3

else:
    Instrucao-c1
    Instrucao-c2

Instrucao4
Instrucao5
Instrucao8
```

As instruções c1 e c2 serão executadas apenas se a condição for **False**





Seleção simples



Seleção composta

Duas atividades da aula anterior

Menor valor entre 4 números

```
def f1(a: float, b: float, c:float, d:float) -> float:
    m1=a
    m2=c;
    if m1>b:
        m1 = b
    if m2>d:
        m2 = d
    if m1>m2:
        return m2
    else:
        return m1
```

```
print(f1(4, -50, 6, -34))
```

```
-50
```

Fatorial de um número inteiro

```
def fff(n: int) -> int:
    if n==0:
        return 1
    else:
        return fff(n-1)*n
```

```
print(fff(5))
```

Exercício: Atribuir conceito

Crie uma função que permite devolver o conceito dada a nota de um aluno

```
def atribuir_conceito(nota: float) -> str:
```

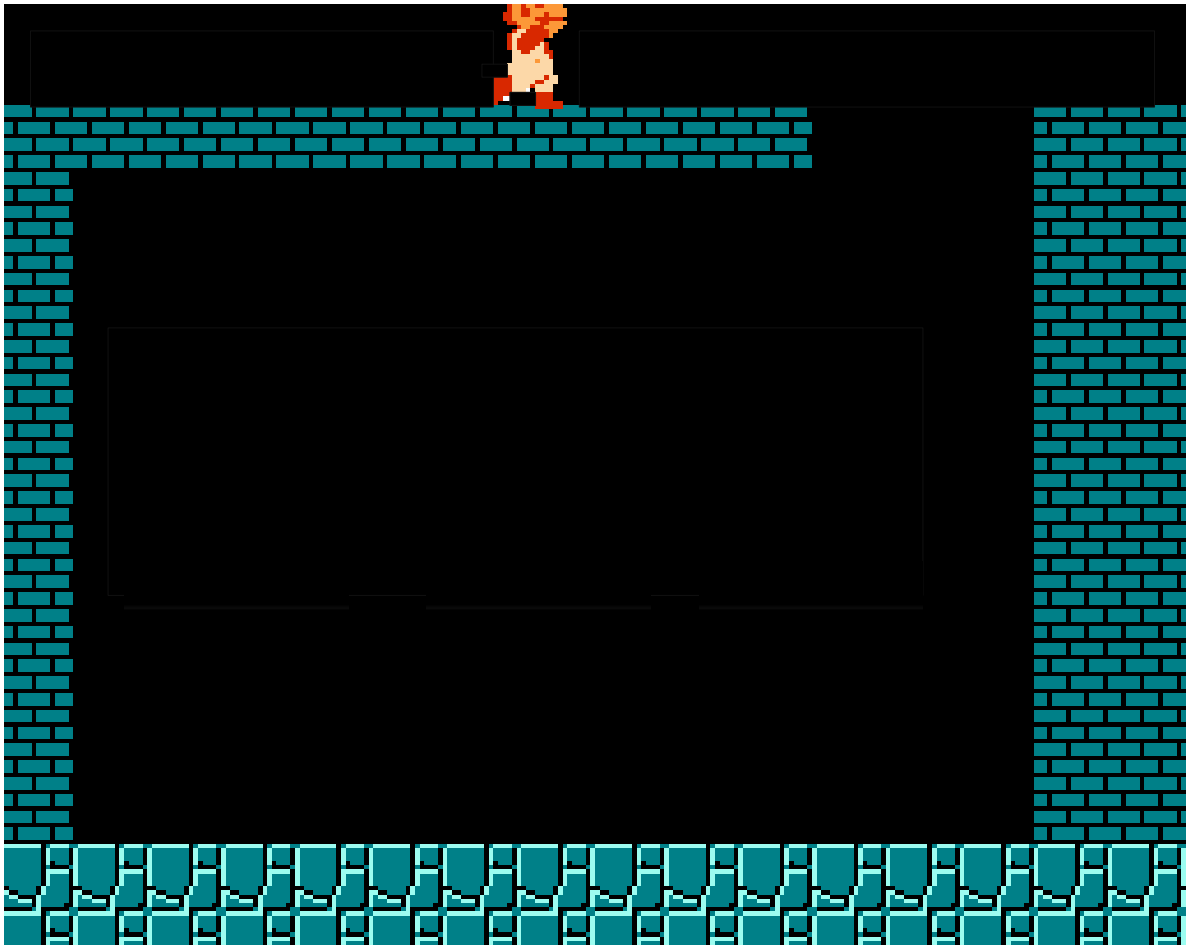
- **A:** nota ≥ 9
- **B:** 7,5 ≤ nota < 9
- **C:** 6 ≤ nota < 7,5
- **D:** 5,0 ≤ nota < 6
- **F:** nota < 5,0

```python
def atribuir_conceito(nota: float) -> str:
    conceito = ""

    if nota>=9:
        conceito = "A"
    else:
        if nota>=7.5:
            conceito = "B"
        else:
            if nota>=6:
                conceito = "C"
            else:
                if nota>=5:
                    conceito = "D"
                else:
                    conceito = "F"
    return conceito
```

elif é a forma abreviada de **else+if**

```
def atribuir_conceito(nota: float) -> str:
    conceito = ""

    if nota>=9:
        conceito = "A"
    elif nota>=7.5:
        conceito = "B"
    elif nota>=6:
        conceito = "C"
    elif nota>=5:
        conceito = "D"
    else:
        conceito = "F"
    return conceito
```

```
def nomeFuncao(var):
```

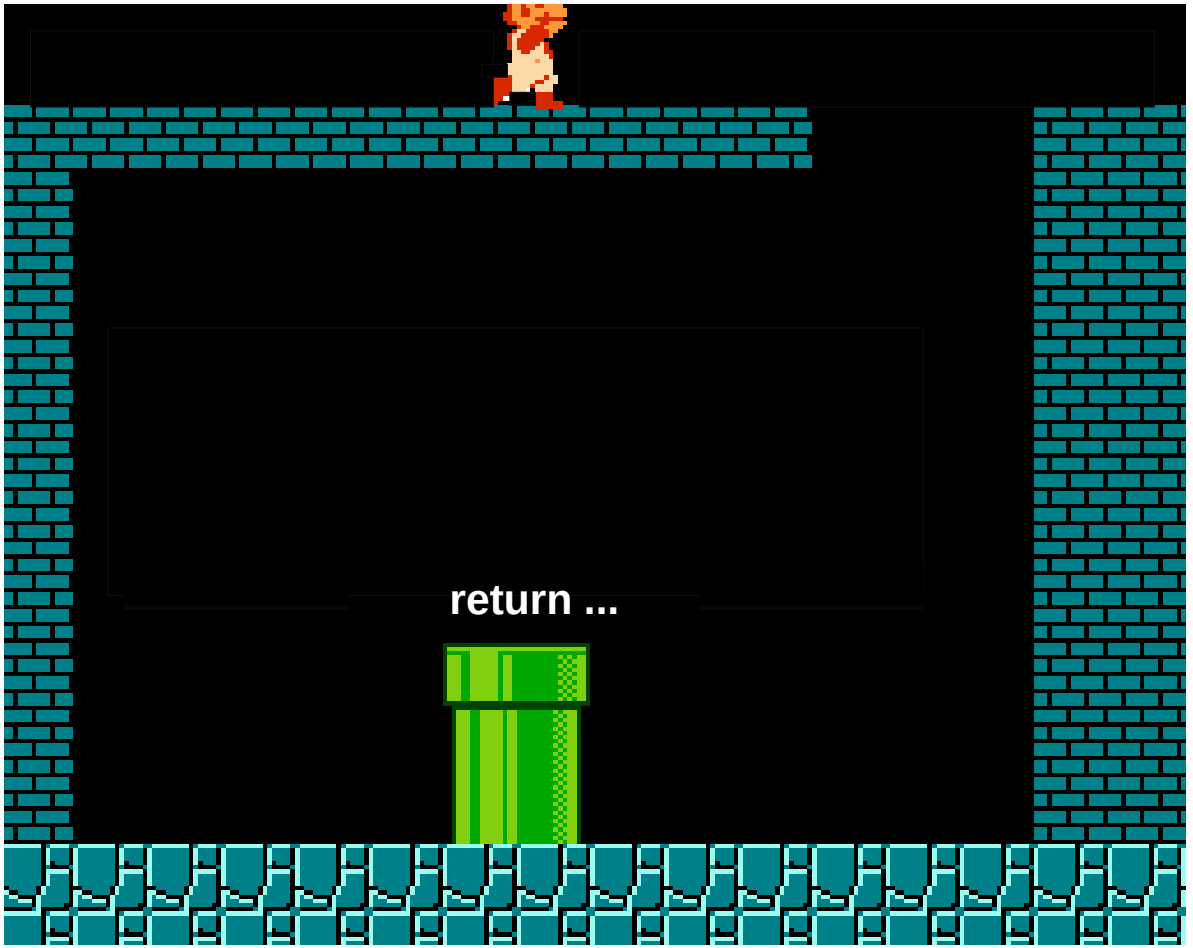


Função/
Método

(c) Super Mario Bros.

```
def nomeFuncao(var):
```

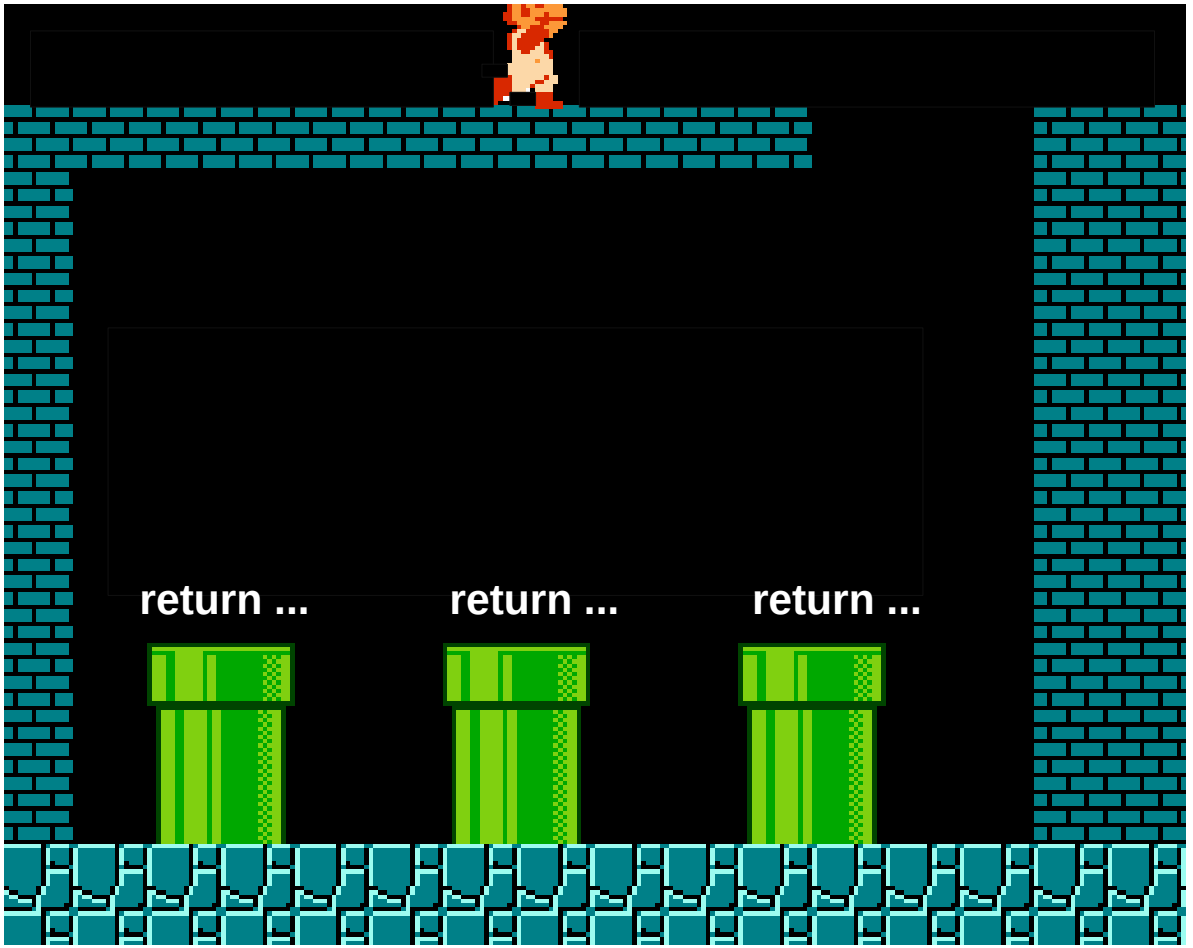
Função/
Método



(c) Super Mario Bros.

```
def nomeFuncao(var):
```

Função/
Método



(c) Super Mario Bros.

```python
def atribuir_conceito(nota: float) -> str:
    if nota>=9:
        return "A"
    if nota>=7.5:
        return "B"
    if nota>=6:
        return "C"
    if nota>=5:
        return "D"
    return "F"
```

Qual seria o resultado de execução do seguinte trecho?

```
1   x = 12
2
3   if x>0:
4       print("positivo")
5   if x%2==0:
6       print("par")
7   if x%3==0:
8       print("ímpar")
```

Qual seria o resultado de execução do seguinte trecho?

```
1   x = 12
2
3   if x>0:
4       print("positivo")
5   if x%2==0:
6       print("par")
7   if x%3==0:
8       print("ímpar")
```

```
positivo
par
ímpar
```

Qual seria o resultado de execução do seguinte trecho?

```
1    x = 12
2
3    if x>0:
4        print("positivo")
5    elif x%2==0:
6        print("par")
7    elif x%3==0:
8        print("ímpar")
```

```
positivo
```

Operadores lógicos

Operadores lógicos

Em Python existem 3 operadores lógicos:

- `and`
- `or`
- `not`

A semântica destes operadores é similar ao seu significado em Inglês/Português.

Por exemplo a expressão **`x>0 and x<10`**

é verdadeira somente se x é maior a zero **e** menor do que dez.

Operadores lógicos

Finalmente, o operador not nega uma expressão booleana, assim

```
not x > y
```

é verdadeira
se x>y for falso isto é, se x é **menor ou igual** a y.

Exercício: Risco de doença cardíaca

Podemos usar uma versão simplificada para calcular o risco de doença cardíaca de uma pessoa usando as seguintes regras baseadas na **idade** e no **índice de massa corporal** (IMC):

		idade	
		<45	**>= 45**
IMC	**< 22.0**	baixo	médio
	>= 22.0	médio	alto

Exercício: Risco de doença cardíaca

```
def risco_cardiaco(idade: int, imc: float) -> str:
    if idade < 45:
        if imc < 22:
            risco = "baixo"
        else:
            risco = "médio"
    else:
        if imc < 22:
            risco = "médio"
        else:
            risco = "alto"
    return risco
```

A comparação imc<22
é usada 2 vezes!

Exercício: Risco de doença cardíaca

```
def risco_cardiaco(idade: int, imc: float) -> str:
    jovem = idade < 45
    magro = imc < 22

    if jovem:
        if magro:
            risco = "baixo"
        else:
            risco = "médio"
    else:
        if magro:
            risco = "médio"
        else:
            risco = "alto"
    return risco
```

O resultado de uma expressão booleana pode ser armazenada em uma variável

Exercício: Risco de doença cardíaca

```python
def risco_cardiaco(idade: int, imc: float) -> str:
    jovem = idade < 45
    magro = imc < 22

    if jovem and magro:
        risco = "baixo"
    elif jovem and not magro:
        risco = "médio"
    elif not jovem and magro:
        risco = "médio"
    elif not jovem and not magro:
        risco = "alto"
    return risco
```

Exercício: Risco de doença cardíaca

```
def risco_cardiaco(idade: int, imc: float) -> str:
    jovem = idade < 45
    magro = imc < 22

    if jovem and magro:
        risco = "baixo"
    elif jovem and not magro:
        risco = "médio"
    elif not jovem and magro:
        risco = "médio"
    else:
        risco = "alto"
    return risco
```

Sobre o EP1_7 - Encriptação

1234 → 2345

9092 → 0103

EP1 - Sequencial

Todos os exercícios deveriam ser resolvidos com apenas Sequências simples, operações simples.

- Não precisa utilizar laços!
- Não precisa utilizar condicionais!
- Não precisa utilizar funções sofisticadas de strings!

Pense numa massagem matemática!

EP1 - Sequencial

Suponha o caso menor: **90** deve virar **01**

```
numero = 90

d1 = numero//10
d2 = numero%10

d1 = (d1+1)%10
d2 = (d2+1)%10

print("{}{}".format(d1, d2))

print(f"{d1}{d2}")
```

Sobre o operador módulo

O operador módulo acaba sendo surpreendentemente útil.

Por exemplo, você pode verificar se um número é divisível por outro, se x%y é zero, então x é divisível por y

```
>>> 24 % 1   → 0
>>> 24 % 2   → 0
>>> 24 % 3   → 0
>>> 24 % 4   → 0
>>> 24 % 5   → 4
>>> 24 % 6   → 0
>>> 24 % 7   → 3
>>> 24 % 8   → 0
>>> 24 % 9   → 6
```

Sobre o operador módulo

Este operador pode ser utilizado para extrair o(s) dígito(s) mais à direita de um número.

Por exemplo:

```
>>> 12345 // 10     → 1234
>>> 12345%10        → 5
```

Mantém o digito mais à direita

```
>>> 12345 // 100    → 123
>>> 12345%100       → 45
```

Mantém os 2 digitos mais à direita

Atividade em aula

Questão 1 (a)

```
def total1(c: str) -> int:
    t = 4;
    if c=="A":
        t = t-0
    if c=="B":
        t = t-1
    if c=="C":
        t = t-2
    if c=="D":
        t = t-3
    else:
        t = t-4
    return t
```

```
print(total1("A"))
print(total1("B"))
```

```
0
-1
```

Questão 1 (b)

```
def total2(c: str) -> int:
    if c=="A":
        t = 4
    if c=="B":
        t = 3
    if c=="C":
        t = 2
    if c=="D":
        t = 1
    else:
        t = 0
    return t
```

```
print(total2("A"))
print(total2("B"))
```

```
0
0
```

Questão 1 (c)

```
def conceito(t: float, f: int) -> str:
    var = ""
    if t>=5 and t<=10:
        var = "aprovado"
    if f>=6:
        var = "reprovado"
    else:
        var = "aprovado"
    return var
```

```
print( conceito(3, 0) )
print( conceito(6, 6) )
```

```
aprovado
reprovado
```

Questão 1 (d)

```
def troca(w: int, q: int) -> int:
    if w>q:
        w = troca(q, w)
    return w
```

```
print(troca(3,1))
print(troca(-1,1))
```

```
1
-1
```

Questão 1 (e)

```
def mat(x: int) -> int:
    if x%10>=5:
        return (x//10)%10
    else:
        return x%10
```

```
print( mat(678) )
print( mat(1234) )
```

```
7
4
```

Bônus

Crie uma função em que, dados 3 números como parâmetros, permita verificar se a soma de quaisquer par de números gera a soma do terceiro número.

Sua função deve devolver True ou False:

```
def verifica_soma(a: int, b: int, c: int) -> bool:
    return a+b==c or a+c==b or b+c==a
```