# Pensamento Computacional

U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

# Objetivos

- Expressões booleanas
  - Tipo bool;
  - Operadores relacionais;
  - Operadores lógicos;
  - Propriedades.
- Execução condicional
  - If statement
  - If-else
  - If-elif-else
- Expressões condicionais

# Expressões booleanas

- Uma **expressão booleana** é uma expressão que ou é Verdadeira (True) ou é Falsa (False).

```
>>> n = 5       # this IS NOT a boolean expression!
>>> n == 5      # this IS a boolean expression!
True
>>> 6 == n      # this is another boolean expression.
False
```

- `True` e `False` são valores especiais que pertencem ao tipo `bool`.

# Expressões booleanas

- Valores booleanos podem ser guardados em variáveis:

```
>>> isEven = n%2==0
```

- Podem ser convertidos para string:

```
>>> str(isEven)
'False'
```

- Ou inteiros:

```
>>> int(False)   # 0
>>> int(True)    # 1
```

# Expressões booleanas



Valores nulos ou vazios convertem para `False`:

```
>>> bool(0)          # False
>>> bool(0.0)        # False
>>> bool('')         # False
>>> bool([])         # False
```

Outros valores convertem para `True`:

```
>>> bool(1)          # True
>>> bool('False')    # True (surprise!)
>>> bool([False])    # True (surprise?)
```

# Operadores relacionais e lógicos

U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

- **Operadores relacionais** produzem resultados booleanos:

```
x == y       # x is equal to y
x != y       # x is not equal to y
x > y        # x is greater than y
x < y        # x is less than y
x >= y       # x is greater than or equal to y
x <= y       # x is less than or equal to y
x < y < z    # x is less than y and y is less than z
              (cool!)
```

# Operadores relacionais e lógicos

U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

- Há três **operadores lógicos**: `and`, `or`, `not`.

```
x>=0  and  x<10          #  x  is  between  0  and  10
(exclusive)

0<=x and x<10     # same thing

x==0 or not isEven and y/x>1
```

# Propriedades

- Lembrem-se destas propriedades:

```
x == y   <=>   not (x != y)   <=>   y == x
x != y   <=>   not (x == y)   <=>   y != x
x > y    <=>   not (x <= y)   <=>   y < x
x <= y   <=>   not (x > y)    <=>   y >= x
```

- E destas (onde A, B, C são booleanos):

```
not (not A)        <=>   A
not (A and B)      <=>   (not A) or (not B)
not (A or B)       <=>   (not A) and (not B)
A or B             <=>   B or A
A and B            <=>   B and A
A or (B and C)     <=>   (A or B) and (A or C)
A and (B or C)     <=>   (A and B) or (A and C)
```

# Regras de precedência

- Aritméticas> relacionais > `not` > `and` > `or`.

```
x<=1+2*y**3 or n!=0 and not 1/n<=y

(x<=1+2*y**3) or (n!=0 and not 1/n<=y)

(x<=(1+2*y**3)) or ((n!=0) and (not 1/n<=y))

(x<=(1+(2*y**3))) or ((n!=0) and (not (1/n<=y)))

(x<=(1+(2*(y**3)))) or ((n!=0) and (not ((1/n)<=y)))
```

# Exemplos

```
23 < 4                                  #->  False
5 < 10 < 20                             #->  True
5 == 3 + 2                              #->  True
15 > 5*2 and 10 - 2 == 9                #->  False
```

- Os operadores relacionais também funcionam com strings.

```
'sky' > 'clouds'                        #-> True
'Sky' > 'clouds'                        #-> False
'8' > '5'                               #-> True
'123' > '4'                             #-> False
```

# Avaliação curto-circuito (short-circuit)

- Os operadores `and` e `or` só avaliam o 2º operando se necessário!

```
A and B    # if A is false then A, otherwise B
A or B     # if A is true then A, otherwise B
```

- A isto chama-se de **avaliação curto-circuito**.
- Pode ser muito útil:

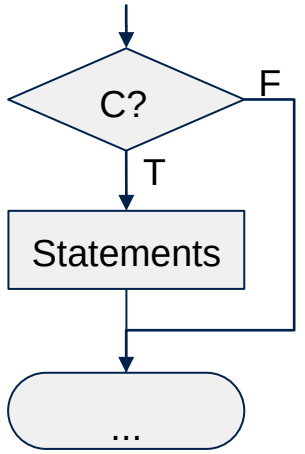
```
1/n>2 and n!=0    # ZeroDivisionError if n==0
n!=0 and 1/n>2    # False if n==0, 1/n not evaluated
n==0 or 3/n<4     # True if n==0, 3/n not evaluated
```

Note que a ordem dos operandos é importante!

# Execução condicional

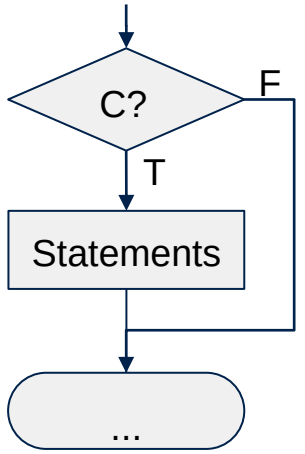
- **As expressões condicionais** permitem que o programa verifique certas condições e altere o seu comportamento de adequadamente.
- A sua forma mais simples é a expressão `if` :

```
if condition:
    suite_of_statements
...
```



# Execução condicional

```
if condition:
    suite_of_statements
...
```



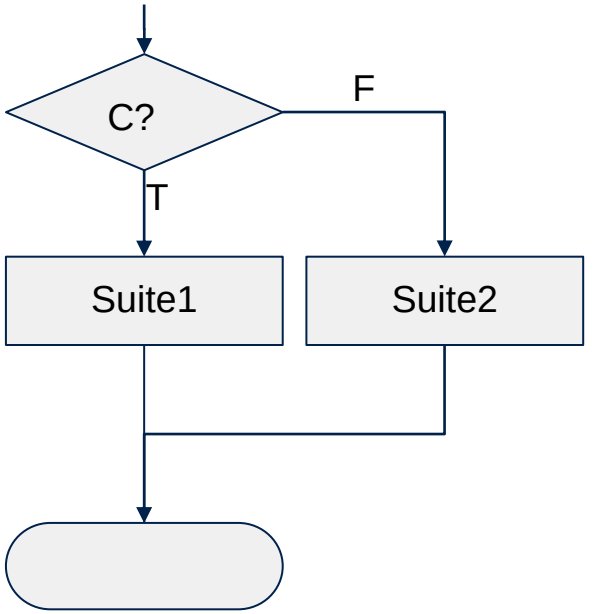

- A *condition* deve ser uma expressão booleana.
- A *suite of statements* é executada se a condição for verdadeira. Se não, a execução continua após estas estas declarações (*statements*).
- O conjunto de declarações deve ter uma ou mais declarações *indentadas*.

# Execução condicional

- Um outro formato da declaração `if` é a de execução alternativa, na qual existem dois caminhos alternativos e a condição determina qual é que é executado.

```
if x%2 == 0:
    print('x is even')
else:
    print('x is odd')

#END
```

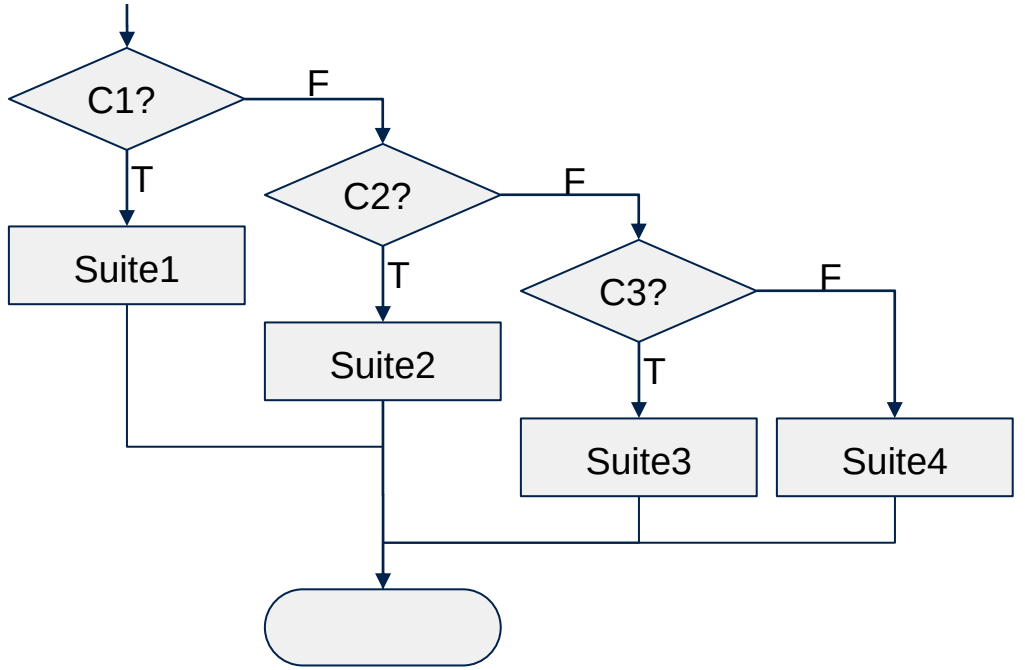


# Execução condicional

- Por vezes, existem mais do que dois caminhos alternativos. Neste caso, precisamos de adicionar mais ramos à expressão condicional.

```
if x < 10:
    mark = 'Poor'
elif x < 13:
    mark = 'Reasonable'
elif x < 17:
    mark = 'Good'
else:
    mark = 'Excelent'

print(mark)
```



# Semântica das Expressões Condicionais

- Quais as condições que ativam cada conjunto de declarações?

```
if C1:
    Suite1          ← Suite1 is executed iff C1
elif C2:
    Suite2          ← Suite2 is executed iff ¬C1∧C2
elif C3:
    Suite3          ← Suite3 is executed iff ¬C1∧¬C2∧C3
else:
    Suite4          ← Suite4 is executed iff ¬C1∧¬C2∧¬C3

Rest                ← is always executed
```

# Exemplos: if-elif-else

```
num = int(input('num? '))

if num < 10:
    print(num, 'is less than 10.')
elif num > 10:
    print(num, 'is greater than 10.')
else:
    print(num, 'is equal to 10!')
```

```
word = input('Word: ')

if word < 'banana':
    print(word, 'comes before banana.')
elif word > 'banana':
    print(word, 'comes after banana.')
else:
    print('Word is "banana"!')
```

# Declarações condicionais combinadas *(nested)*

U.PORTO
FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

- As declarações condicionais podem ser combinadas entre si.

```
if y > 0:
    if x > 0:
        quadrant = 1
    else:
        quadrant = 2
else:
    if x < 0:
        quadrant = 3
    else:
        quadrant = 4
```

- Embora a indentação permita facilitar a criação e compreensão da estrutura, expressões combinadas alagadas podem tornar-se difíceis de ler! É importante encontrar formas de simplificação.

# Expressão Condicional

U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

- Python também inclui uma expressão condicional, baseada num operador ternário:

```
expression1 if condition else expression2
```

- Usa as *keywords* if e else, mas é uma *expressão*!
- A condição é a primeira a ser avaliada.
- Se *true*, então expression1 é avaliada e é o resultado.
- Se *false*, então expression2 é avaliada e é o resultado.

"

```
n = int(input("number? "))
msg = "odd" if n%2!=0 else "even"
print(n, "is", msg)
```

# Pensamento Computacional

2022/23

TPC: exercícios propostos ficha 4