Faça um programa que peça 10 números inteiros, calcule e mostre a quantidade de números pares e a quantidade de números ímpares.

Tentativa 1

Solução correta!

```
n = 1
P = 0
I = 0
while n <= 10:
    a = int(input())
    n = n + 1
    if a % 2 == 0:
        a = P
        P = P + 1
    else:
        a = I
        I = I + 1

print("A qtd de números pares é: ", P)
print("A qtd de números ímpares é: ", I)
```

Tentativa 1

Solução correta! Mas, pode ficar melhor =)

```
n = 1
P = 0
I = 0
while n <= 10:
    a = int(input())
    n = n + 1
    if a % 2 == 0:
        a = P
        P = P + 1
    else:
        a = I
        I = I + 1

print("A qtd de números pares é: ", P)
print("A qtd de números ímpares é: ", I)
```

Tentativa 1

Solução correta! Mas, pode ficar melhor =)

```
n = 1
P = 0
I = 0
while n <= 10:
    a = int(input())
    n = n + 1
    if a % 2 == 0:
        a = P
        P = P + 1
    else:
        a = I
        I = I + 1

print("A qtd de números pares é: ", P)
print("A qtd de números ímpares é: ", I)
```

Tentativa 1.1

Solução correta!

```
n = 1
P = 0
I = 0
while n <= 10:
    a = int(input())
    n = n + 1
    if a % 2 == 0:
        P = P + 1
    else:
        I = I + 1

print("A qtd de números pares é: ", P)
print("A qtd de números ímpares é: ", P)
```

Tentativa 2

Solução incorreta. Por que?

```
a = 10
i = 1
P = 0
I = 0
while i <= 10:
    numero = int(input("Insira um número"))
        i = i + 1
        if (numero // 2 == 0):
        P = P + 1
            else:
            I = I + 1

print("Números pares: ", P, " números ímpares: ", I)
```

Tentativa 2

Solução incorreta. Por que?

```
a = 10
i = 1
P = 0
I = 0
while i <= 10:
    numero = int(input("Insira um número"))
    |   i = i + 1
    |   if (numero // 2 == 0):
    |   P = P + 1
    |   |   else:
    |   |   I = I + 1

print("Números pares: ", P, " números ímpares: ", I)
```

Erros de indentação, para o while e para o if-else.

`numero // 2 == 0` não identifica os números pares.

Para que serve `a`?

Tentativa 2.1

Solução correta.

```
i = 1
P = 0
I = 0
while i <= 10:
    numero = int(input("Insira um número"))
    i = i + 1
    if (numero % 2 == 0):
        P = P + 1
    else:
        I = I + 1

print("Números pares: ", P, " números ímpares: ", I)
```

Tentativa 3

Solução correta.

```
numeros = int(input())
i = 1
par = 0
impar = 0
if (numeros % 2 == 0):
    par = 1
else:
    impar = 1

while i < 10:
    proximo_numero = int(input())
    i = i + 1
    if (proximo_numero % 2 == 0):
        par = par + 1
    else:
        impar = impar + 1

print("Quantidade de números pares: ", par)
print("Quantidade de números ímpares é: ", impar)
```

Tentativa 3

Solução correta. Mas, pode ficar melhor =)

```
numeros = int(input())
i = 1
par = 0
impar = 0
if (numeros % 2 == 0):
    par = 1
else:
    impar = 1

while i < 10:
    proximo_numero = int(input())
    i = i + 1
    if (proximo_numero % 2 == 0):
        par = par + 1
    else:
        impar = impar + 1

print("Quantidade de números pares: ", par)
print("Quantidade de números ímpares é: ", impar)
```

Tentativa 3

Solução correta. Mas, pode ficar melhor =)

```
numeros = int(input())
i = 1
par = 0
impar = 0
if (numeros % 2 == 0):
    par = 1
else:
    impar = 1

while i <= 10:
    proximo_numero = int(input())
    i = i + 1
    if (proximo_numero % 2 == 0):
        par = par + 1
    else:
        impar = impar + 1

print("Quantidade de números pares: ", par)
print("Quantidade de números ímpares é: ", impar)
```

Tentativa 3.1

Solução correta!

```
numeros = int(input())
i = 1
par = 0
impar = 0

while i <= 10:
    proximo_numero = int(input())
    i = i + 1
    if (proximo_numero % 2 == 0):
        par = par + 1
    else:
        impar = impar + 1

print("Quantidade de números pares: ", par)
print("Quantidade de números ímpares é: ", impar)
```

Tentativa 4

Solução correta!

```
n = 1
numero_par = 0
while n <= 10:
    numero = int(input("Digite o número"))
    n = n + 1
    if (numero % 2 == 0):
        numero_par = numero_par + 1

print(numero_par, "número(s) par(es) e ", 10-numero_par, "ímpares")
```

Tentativa 5

Solução incorreta. Por que?

```
x = 1
imp = 0
par = 0
while x <= 10:
    n = int(input())
    x = x + 1
    if n % 2 == 0:
        par = par + 1
    if n % 0 != 0:
        imp = imp + 1

print(par)
print(imp)
```

Tentativa 5

Solução incorreta. Por que?

```
x = 1
imp = 0
par = 0
while x <= 10:
    n = int(input())
    x = x + 1
    if n % 2 == 0:
        par = par + 1
    if n % 0 != 0:
        imp = imp + 1

print(par)
print(imp)
```

Erro `if n % 0 != 0:`
Colocou **`if-if`**, poderia ser **`if-else`**.

Tentativa 6

Solução incorreta. Por que?

```
n = 1
while (n <= 10):
    a = int(input("Digite o seu número:"))
    n = n + 1
for a in range(n):
    if a % 2 == 0:
        print(a, "é par")
    else:
        print(a, "é ímpar")
```

Tentativa 6

Solução incorreta. Por que?

```
n = 1
while (n <= 10):
    a = int(input("Digite o seu número:"))
    n = n + 1
for a in range(n):
    if a % 2 == 0:
        print(a, "é par")
    else:
        print(a, "é ímpar")
```

Vários erros.

Por exemplo, como vamos verificar se a é par ou ímpar para os 10 números digitados?

O exercício pede para mostrar a quantidade de números pares e ímpares.

Tentativa 7

Solução correta!

```
pares = 0
impares = 0
for numero in range(1,11):
    inteiro = int(input("Insira um número inteiro:"))
    if inteiro % 2 == 0:
        pares = pares + 1
    else:
        impares = impares + 1

print("A quantidade de números pares é: ", pares)
print("A quantidade de números ímpares é: ", impares)
```

Tentativa 8

Solução incorreta. Por que ?

```
n = int(input())
for in range(n, n+11):
    if (n % 2 == 0):
        print("número par: ", n)
    else:
        print("número ímpar: ", n)
```

Tentativa 8

Solução incorreta. Por que ?

```
n = int(input())
for in range(n, n+11):
    if (n % 2 == 0):
        print("número par: ", n)
    else:
        print("número ímpar: ", n)
```

O comando for está incorreto. `for variavel in range(n, n+11):`

Cadê a entrada dos 10 números?

Cadê a quantidade de números pares e ímpares?

Tentativa 9

Solução incorreta. Por que?

```
for (N = 1, N <= 10):
    int(input("digite número"))
if (N/2 == 0):
    par = par + 1
    N = N + 1
else:
    impar = impar + 1
```

Vários erros. Por exemplo, o comando `for` está incorreto.

Tentativa 10

Solução incorreta. Por que?

```
n = int(input("Digite 10 números:"))
for i in range(1, n+1):
    print(i)
for j in range(n // 2 == 0):
    print(j)
for k in range(n // 2 != 0):
    print(k)
```

Vários erros. Por exemplo, cadê os 10 números?

O comando `for` está incorreto.

Tentativa 11

Solução incorreta. Por que?

```
n1 = int(input("Digite 10 números inteiros:"))
n2 = int(input())
n3 = int(input())
 ⋮
n10 = int(input())
numero = 2
if (n1 … n10 % numero == 0):
    print("É par", n1 … n10)
else:
    print("Não é par", n1 … n10)
```

Python não entende “…” como na língua portuguesa. Não funciona.

Tentativa 12

Solução incorreta. Por que?

```
a1 = int(input("Digite o 1o número:"))
a2 = int(input("Digite o 2o número:"))
a3 = int(input("Digite o 3o número:"))
a4 = int(input("Digite o 4o número:"))
a5 = int(input("Digite o 5o número:"))
a6 = int(input("Digite o 6o número:"))
a7 = int(input("Digite o 7o número:"))
a8 = int(input("Digite o 8o número:"))
a9 = int(input("Digite o 9o número:"))
a10 = int(input("Digite o 10o número:"))

n = numero + 1
z = numero

if (numero % 2 == 0):
    print("A quantidade de números pares é: ", n)
if (numero % 2 != 0):
    print("A quantidade de números ímpares é: ", z)
```

Tentativa 13

Solução correta!

```
numero = 10
pares = 0
for x in range(numero):
    n = int(input("Digite um número:"))
    if (n % 2 == 0):
        pares = pares + 1

print(pares, "números pares")
print((numero - pares), "números ímpares")
```

Tentativa 13

Solução correta!

```
numero = 10
pares = 0
for x in range(numero):
    n = int(input("Digite um número:"))
    if (n % 2 == 0):
        pares = pares + 1

print(pares, "números pares")
print((numero - pares), "números ímpares")
```

Tentativa 13

Solução correta!

```
pares = 0
for x in range(10):
    n = int(input("Digite um número:"))
    if (n % 2 == 0):
        pares = pares + 1

print(pares, "números pares")
print((10 - pares), "números ímpares")
```

Tentativa 14

Solução “correta”.

```
a1 = int(input())
a2 = int(input())
a3 = int(input())
a4 = int(input())
a5 = int(input())
a6 = int(input())
a7 = int(input())
a8 = int(input())
a9 = int(input())
a10 = int(input())
impares = 0
pares = 0

if (a1 % 2 == 0):
      pares = pares + 1
else:
      impares = impares + 1
if (a2 % 2 == 0):
      pares = pares + 1
else:
      impares = impares + 1
if (a3 % 2 == 0):
      pares = pares + 1
else:
      impares = impares + 1
if (a4 % 2 == 0):
      pares = pares + 1
else:
      impares = impares + 1
```

```
if (a5 % 2 == 0):
      pares = pares + 1
else:
      impares = impares + 1
if (a6 % 2 == 0):
      pares = pares + 1
else:
      impares = impares + 1
if (a7 % 2 == 0):
      pares = pares + 1
else:
      impares = impares + 1
if (a8 % 2 == 0):
      pares = pares + 1
else:
      impares = impares + 1
if (a9 % 2 == 0):
      pares = pares + 1
else:
      impares = impares + 1
if (a10 % 2 == 0):
      pares = pares + 1
else:
      impares = impares + 1

print("Você digitou ",pares, "números pares.")
print("Você digitou ",impares, "números ímpares.")
```

Algoritmos e Programação de Computadores

Strings

Profa. Sandra Avila
Instituto de Computação (IC/Unicamp)

MC102, 5 Abril, 2019

Agenda

- Strings
  - Operações
  - Funções
  - Métodos
- Exercícios

Strings

- Strings em Python são listas **imutáveis** de caracteres.
- Strings são representadas por sequências de caracteres entre aspas simples `'` ou entre aspas duplas `"`.

```
a = "26 de Abril tem prova."
a
'26 de Abril tem prova.'
b = 'Fizeram a atividade conceitual?'
b
'Fizeram a atividade conceitual?'
c = "Que vida \"fácil\""
c
'Que vida "fácil"'
```

Strings

- Strings em Python são listas **imutáveis**, portanto pode-se acessar posições de uma string de forma usual.

```
a = "26 de Abril tem prova."
a[0]
'2'
a[0] = "1"
```

```
---------------------------------------------------------------
TypeError                 Traceback (most recent call last)
<ipython-input-13-9ab1dda42293> in <module>()
----> 1 a[0] = "1"

TypeError: 'str' object does not support item assignment
```

Strings

- O caractere `'\n'` pode fazer parte de uma string e ele só causa a mudança de linha no comando `print`.

```
a = 'Fizeram\na\natividade\nconceitual?'
a
'Fizeram\na\natividade\nconceitual?'
```

```
a = 'Fizeram\na\natividade\nconceitual?'
print(a)
Fizeram
a
atividade
conceitual?
```

Strings: Operações, Funções e Métodos

- O operador + concatena 2 strings, e o operador * repete a concatenação (como em listas).

```
a = "26 de Abril tem prova."
b = 'Fizeram a atividade conceitual?'
a + b
'26 de Abril tem prova.Fizeram a atividade conceitual?'
```

```
b = 'Fizeram a atividade conceitual?\n'
print(3*b)
Fizeram a atividade conceitual?
Fizeram a atividade conceitual?
Fizeram a atividade conceitual?
```

Strings como Listas

- Strings podem ser processadas como listas, podendo por exemplo ter seus elementos percorridos num laço **for**.
- Exemplo: Ler uma string e imprimir a inversa.

```
string = input("Digite um texto: ")
inversa = " "
for x in string:
    inversa = x + inversa
print(inversa)
```

Strings: Operações, Funções e Métodos

- A função **slice** (fatiar) devolve a string entre duas posições dadas.
- Pode-se fatiar (slice) strings usando `[inicio:fim-1:passo]`.

```
a = "20 de Abril tem prova."
a[6:11]
'Abril'
a[6:11:2]
'Arl'
a[::-1]
'.avorp met lirbA ed 02'
```

- A string vazia é representada como `' '` ou `" "`.

Strings: Operações, Funções e Métodos

- O método `strip` retorna uma string sem os brancos e mudança de linhas **no início e no final** de uma string.

```
b = "Fizeram a atividade conceitual?"
b
'\n Fizeram a atividade conceitual? \n'

b.strip()
'Fizeram a atividade conceitual?'
```

Strings: Operações, Funções e Métodos

- O operador **in** verifica se uma **substring** é parte de uma outra string.

```
"atividade" in "Fizeram a atividade conceitual?"
True

"idade" in "Fizeram a atividade conceitual?"
True

"Abril" in "Fizeram a atividade conceitual?"
False
```

Strings: Operações, Funções e Métodos

- O método `find` retorna onde a substring começa na string.

```
a = "Fizeram a atividade conceitual?"
a.find("atividade")
10

a.find("abril")
-1
```

- O método `find` retorna `-1` quando a substring não ocorre na string.

Strings: Operações, Funções e Métodos

- O método `split(sep)` separa uma string usando **sep** como separador. Retorna uma lista das substrings.

```
numeros = "1; 2 ;  3"
numeros.split(";")
['1', ' 2 ', '  3']

a = "Fizeram a atividade conceitual?"
a.split()
['Fizeram', 'a', 'atividade', 'conceitual?']
```

- Podem haver substrings vazias no retorno de `split()`.

Strings: Operações, Funções e Métodos

- O método `replace` serve para trocar **todas** as ocorrências de uma substring por outra em uma string.

```
a = "Fizeram a atividade conceitual?"
a.replace("conceitual","teórica")
'Fizeram a atividade teórica?'

a = "Fizeram a atividade conceitual?"
a.replace("conceitual","")
'Fizeram a atividade ?'
```

Strings: Operações, Funções e Métodos

- Podemos usar a função `list` para transformar uma string em uma lista onde os itens da lista correspondem aos caracteres da string.

```
numeros = "1; 2 ;  3"
list(numeros)
['1', ';', ' ', '2', ' ', ';', ' ', ' ', '3']

list("atividade")
['a', 't', 'i', 'v', 'i', 'd', 'a', 'd', 'e']
```

Strings: Operações, Funções e Métodos

- O método `join` recebe como parâmetro uma sequência ou lista, e retorna uma string com a concatenação dos elementos da sequência/lista.

```
l = list("atividade")
l
['a', 't', 'i', 'v', 'i', 'd', 'a', 'd', 'e']

"".join(l)
'atividade'
```

Exercícios

Exemplo: Contador de Palavras

- Faça um programa que conta o número de palavras em um texto.

Exemplo: Contador de Palavras

- Faça um programa que conta o número de palavras em um texto.
  - Primeiramente removemos do texto todos os sinais de pontuação.

```
texto = input("Digite um texto: ")
pontuacao = [".", ",", ":", ";", "!", "?"]

# remove os sinais de pontuação
for p in pontuacao:
    texto = texto.replace(p," ")
```

Exemplo: Contador de Palavras

- Faça um programa que conta o número de palavras em um texto.
  - Depois usamos a função `split` para separar as palavras.

```
texto = input("Digite um texto: ")
pontuacao = [".", ",", ":", ";", "!", "?"]

# remove os sinais de pontuação
for p in pontuacao:
    texto = texto.replace(p," ")

# split devolve lista com palavras como itens
numero_palavras = len(texto.split())
print("Número de palavras:", numero_palavras)
```

Exercício: Palíndromo

- Faça um programa que lê uma string e imprime “Palíndromo” caso a string seja um palíndromo e “Não é palíndromo” caso não seja.
    - Assuma que a entrada não tem acentos e que todas as letras são minúsculas.

- Obs: Um *palíndromo* é uma palavra ou frase, que é igual quando lida da esquerda para a direita ou da direita para a esquerda (espaços em brancos são descartados).
    - Exemplos de palíndromo: “ovo”, “reviver”, “mega bobagem”, “anotaram a data da maratona”

Referências & Exercícios

- Os slides dessa aula foram baseados no material de MC102 do Prof. Eduardo Xavier (IC/Unicamp)
- https://wiki.python.org.br/ExerciciosComStrings: 14 exercícios =)
- https://panda.ime.usp.br/pensepy/static/pensepy/08-Strings/strings.html