MAC 110 — Introdução à Ciência da Computação

Aula 11

Nelson Lago

BMAC – 2024

Exercício — sistema de *login*

Imagine um sistema de *login* com três usuários:

- **Alan Turing** — UID **`turing`**, senha **`tmachine`**
- **Ada Lovelace** — UID **`llace`**, senha **`anengine`**
- **Grace Hopper** — UID **`hopper`**, senha **`business`**

Crie um sistema que lê o UID (*login*) e a senha do usuário e, se os dados estiverem corretos, escreve “Bem-vindo, [nome]!”; caso contrário, o sistema escreve “Login ou senha incorreto”.

Exercício — sistema de *login*

Exercício — sistema de *login*

```
uid = input("username: ")
senha = input("senha: ")
```

Exercício — sistema de *login*

```
uid = input("username: ")
senha = input("senha: ")









if user_ok:
  print("Bem-vindo, {}!".format(nome))
else:
  print("Login ou senha incorreto")
```

Exercício — sistema de *login*

```
uid = input("username: ")
senha = input("senha: ")
user_ok = False









if user_ok:
  print("Bem-vindo, {}!".format(nome))
else:
  print("Login ou senha incorreto")
```

Exercício — sistema de *login*

```
uid = input("username: ")
senha = input("senha: ")
user_ok = False
if uid == "turing" and senha == "tmachine":
    user_ok = True
    nome = "Alan Turing"









if user_ok:
  print("Bem-vindo, {}!".format(nome))
else:
  print("Login ou senha incorreto")
```

Exercício — sistema de *login*

```
uid = input("username: ")
senha = input("senha: ")
user_ok = False
if uid == "turing" and senha == "tmachine":
    user_ok = True
    nome = "Alan Turing"
elif uid == "llace" and senha == "anengine":
    user_ok = True
    nome = "Ada Lovelace"
elif uid == "hopper" and senha == "business":
    user_ok = True
    nome = "Grace Hopper"
if user_ok:
  print("Bem-vindo, {}!".format(nome))
else:
  print("Login ou senha incorreto")
```

Exercício — sistema de *login*

```
uid = input("username: ")
senha = input("senha: ")
nome = ""
if uid == "turing" and senha == "tmachine":

    nome = "Alan Turing"
elif uid == "llace" and senha == "anengine":

    nome = "Ada Lovelace"
elif uid == "hopper" and senha == "business":

    nome = "Grace Hopper"
if nome != "":
  print("Bem-vindo, {}!".format(nome))
else:
  print("Login ou senha incorreto")
```

Exercício — sistema de *login*

```
uid = input("username: ")
senha = input("senha: ")

nome = ""
if uid == "turing" and senha == "tmachine":
    nome = "Alan Turing"
elif uid == "llace" and senha == "anengine":
    nome = "Ada Lovelace"
elif uid == "hopper" and senha == "business":
    nome = "Grace Hopper"

if nome != "":
  print("Bem-vindo, {}!".format(nome))
else:
  print("Login ou senha incorreto")
```

Exercício — sistema de *login*

```
def main():
    uid = input("username: ")
    senha = input("senha: ")
    nome = checa_login(uid, senha)
    if nome == "":
        print("Login ou senha incorreto")
    else:
        print("Bem-vindo, {}!".format(nome))
```

Exercício — sistema de *login*

```python
def main():
    uid = input("username: ")
    senha = input("senha: ")
    nome = checa_login(uid, senha)
    if nome == "":
        print("Login ou senha incorreto")
    else:
        print("Bem-vindo, {}!".format(nome))

def checa_login(uid, senha):
    nome = ""
    if uid == "turing" and senha == "tmachine":
        nome = "Alan Turing"
    elif uid == "llace" and senha == "anengine":
        nome = "Ada Lovelace"
    elif uid == "hopper" and senha == "business":
        nome = "Grace Hopper"
    return nome
```

Exercício — sistema de *login*

```
def main():
    uid = input("username: ")
    senha = input("senha: ")
    nome = checa_login(uid, senha)
    if nome == "":
        print("Login ou senha incorreto")
    else:
        print("Bem-vindo, {}!".format(nome))

def checa_login(login, pwd):
    name = ""
    if login == "turing" and pwd == "tmachine":
        name = "Alan Turing"
    elif login == "llace" and pwd == "anengine":
        name = "Ada Lovelace"
    elif login == "hopper" and pwd == "business":
        name = "Grace Hopper"
    return name
```

Exercício — sistema de *login*

Modifique o código anterior para que o programa faça saudações diferentes em função do horário:

- "Bom dia, [nome]!" (das 6h às 12h59)
- "Boa tarde, [nome]!" (das 13h às 18h59)
- "Boa noite, [nome]!" (após as 19h ou antes das 6h)

Para saber o horário atual:

```
import time
hora = int(time.strftime('%H'))
```

Exercício — sistema de *login*

```
def checa_login(uid, senha):
    nome = ""
    if uid == "turing" and senha == "tmachine":
        nome = "Alan Turing"
    elif uid == "llace" and senha == "anengine":
        nome = "Ada Lovelace"
    elif uid == "hopper" and senha == "business":
        nome = "Grace Hopper"
    return nome
```

Exercício — sistema de *login*

```
def main():
    uid = input("username: ")
    senha = input("senha: ")
    nome = checa_login(uid, senha)
    if nome == "":
        print("Login ou senha incorreto")
    else:
```

Exercício — sistema de *login*

```
def main():
    uid = input("username: ")
    senha = input("senha: ")
    nome = checa_login(uid, senha)
    if nome == "":
        print("Login ou senha incorreto")
    else:
        print(saudacao(nome))
```

Exercício — sistema de *login*

```
import time
def saudacao(nome):
```

Exercício — sistema de *login*

```
import time
def saudacao(nome):
    hora = int(time.strftime('%H'))
```

Exercício — sistema de *login*

```
import time
def saudacao(nome):
    hora = int(time.strftime('%H'))
    if hora >= 6 and hora < 13:
        return "Bom dia, {}!".format(nome)
```

Exercício — sistema de *login*

```
import time
def saudacao(nome):
    hora = int(time.strftime('%H'))
    if hora >= 6 and hora < 13:
        return "Bom dia, {}!".format(nome)
    elif hora >= 13 and hora < 19:
        return "Boa tarde, {}!".format(nome)
    else:
        return "Boa noite, {}!".format(nome)
```

Exercício — sistema de *login*

```
import time
def main():
    nome = checa_login(input("username: "), input("senha: "))
    if nome == "":
        print("Login ou senha incorreto")
    else:
        print(saudacao(nome))

def saudacao(nome):
    hora = int(time.strftime('%H'))
    if hora >= 6 and hora < 13:
        return "Bom dia, {}!".format(nome)
    elif hora >= 13 and hora < 19:
        return "Boa tarde, {}!".format(nome)
    else:
        return "Boa noite, {}!".format(nome)
```

Exercício — sistema de *login*

```
import time
def main():
    nome = checa_login(input("username: "), input("senha: "))
    print(saudacao(nome))

def saudacao(nome):
    if nome == "":
        return "Login ou senha incorreto"
    hora = int(time.strftime('%H'))
    if hora >= 6 and hora < 13:
        return "Bom dia, {}!".format(nome)
    elif hora >= 13 and hora < 19:
        return "Boa tarde, {}!".format(nome)
    else:
        return "Boa noite, {}!".format(nome)
```

Exercício — soma de um trio Pitagórico

Três números inteiros positivos a, b e c, com $a < b < c$, formam um *trio Pitagórico* se $a^2 + b^2 = c^2$. Por exemplo, os números 3, 4 e 5 formam um trio Pitagórico pois $3^2 + 4^2 = 5^2$. Alguns números inteiros positivos podem ser escritos como a soma de um trio Pitagórico. Por exemplo, 12 é um desses números, pois $3 + 4 + 5 = 12$.

Escreva um programa que lê um número inteiro positivo e verifica se ele corresponde à soma de um trio pitagórico. O programa deve imprimir os números que compõem o trio ou uma mensagem informando que o número não é a soma de nenhum trio pitagórico.

Exercício — soma de um trio Pitagórico

Três números inteiros positivos a, b e c, com $a < b < c$, formam um *trio Pitagórico* se $a^2 + b^2 = c^2$. Por exemplo, os números 3, 4 e 5 formam um trio Pitagórico pois $3^2 + 4^2 = 5^2$. Alguns números inteiros positivos podem ser escritos como a soma de um trio Pitagórico. Por exemplo, 12 é um desses números, pois $3 + 4 + 5 = 12$.

Escreva um programa que lê um número inteiro positivo e verifica se ele corresponde à soma de um trio pitagórico. O programa deve imprimir os números que compõem o trio ou uma mensagem informando que o número não é a soma de nenhum trio pitagórico.

```
def pitagorico(a, b, c):

```

Exercício — soma de um trio Pitagórico

Três números inteiros positivos a, b e c, com $a < b < c$, formam um *trio Pitagórico* se $a^2 + b^2 = c^2$. Por exemplo, os números 3, 4 e 5 formam um trio Pitagórico pois $3^2 + 4^2 = 5^2$. Alguns números inteiros positivos podem ser escritos como a soma de um trio Pitagórico. Por exemplo, 12 é um desses números, pois $3 + 4 + 5 = 12$.

Escreva um programa que lê um número inteiro positivo e verifica se ele corresponde à soma de um trio pitagórico. O programa deve imprimir os números que compõem o trio ou uma mensagem informando que o número não é a soma de nenhum trio pitagórico.

```
def pitagorico(a, b, c):
  return a**2 + b**2 == c**2
```

Exercício — soma de um trio Pitagórico

Exercício — soma de um trio Pitagórico

```
def main():
```

Exercício — soma de um trio Pitagórico

```
def main():
  n = int(input("Digite um inteiro positivo: "))
```

Exercício — soma de um trio Pitagórico

```
def main():
  n = int(input("Digite um inteiro positivo: "))
  if achei:
    print("O número {} é a soma do trio pitagórico {}, {} e {}".format(n, a, b, c))
  else:
    print("O número {} não é a soma de nenhum trio pitagórico".format(n))
```

Exercício — soma de um trio Pitagórico

```
def main():
  n = int(input("Digite um inteiro positivo: "))

  achei = False
  while not achei









  if achei:
    print("O número {} é a soma do trio pitagórico {}, {} e {}".format(n, a, b, c))
  else:
    print("O número {} não é a soma de nenhum trio pitagórico".format(n))
```

Exercício — soma de um trio Pitagórico

```
def main():
  n = int(input("Digite um inteiro positivo: "))

  achei = False
  while not achei and a < n:









  if achei:
    print("O número {} é a soma do trio pitagórico {}, {} e {}".format(n, a, b, c))
  else:
    print("O número {} não é a soma de nenhum trio pitagórico".format(n))
```

Exercício — soma de um trio Pitagórico

```
def main():
  n = int(input("Digite um inteiro positivo: "))
  a = 1
  achei = False
  while not achei and a < n:









  if achei:
    print("O número {} é a soma do trio pitagórico {}, {} e {}".format(n, a, b, c))
  else:
    print("O número {} não é a soma de nenhum trio pitagórico".format(n))
```

Exercício — soma de um trio Pitagórico

```
def main():
  n = int(input("Digite um inteiro positivo: "))
  a = 1
  achei = False
  while not achei and a < n:









      a += 1

  if achei:
    print("O número {} é a soma do trio pitagórico {}, {} e {}".format(n, a, b, c))
  else:
    print("O número {} não é a soma de nenhum trio pitagórico".format(n))
```

Exercício — soma de um trio Pitagórico

```
def main():
  n = int(input("Digite um inteiro positivo: "))
  a = 1
  achei = False
  while not achei and a < n:
    b = a +1









      a += 1

  if achei:
    print("O número {} é a soma do trio pitagórico {}, {} e {}".format(n, a, b, c))
  else:
    print("O número {} não é a soma de nenhum trio pitagórico".format(n))
```

Exercício — soma de um trio Pitagórico

```
def main():
  n = int(input("Digite um inteiro positivo: "))
  a = 1
  achei = False
  while not achei and a < n:
    b = a +1
    while not achei and a + b < n:









      a += 1

  if achei:
    print("O número {} é a soma do trio pitagórico {}, {} e {}".format(n, a, b, c))
  else:
    print("O número {} não é a soma de nenhum trio pitagórico".format(n))
```

Exercício — soma de um trio Pitagórico

```
def main():
  n = int(input("Digite um inteiro positivo: "))
  a = 1
  achei = False
  while not achei and a < n:
    b = a +1
    while not achei and a + b < n:




        b += 1

      a += 1

  if achei:
    print("O número {} é a soma do trio pitagórico {}, {} e {}".format(n, a, b, c))
  else:
    print("O número {} não é a soma de nenhum trio pitagórico".format(n))
```

Exercício — soma de um trio Pitagórico

```
def main():
  n = int(input("Digite um inteiro positivo: "))
  a = 1
  achei = False
  while not achei and a < n:
    b = a +1
    while not achei and a + b < n:
      c = n -a -b



        b += 1

      a += 1

  if achei:
    print("O número {} é a soma do trio pitagórico {}, {} e {}".format(n, a, b, c))
  else:
    print("O número {} não é a soma de nenhum trio pitagórico".format(n))
```

Exercício — soma de um trio Pitagórico

```
def main():
  n = int(input("Digite um inteiro positivo: "))
  a = 1
  achei = False
  while not achei and a < n:
    b = a +1
    while not achei and a + b < n:
      c = n -a -b
      achei = pitagorico(a, b, c)

        b += 1

      a += 1

  if achei:
    print("O número {} é a soma do trio pitagórico {}, {} e {}".format(n, a, b, c))
  else:
    print("O número {} não é a soma de nenhum trio pitagórico".format(n))
```

Exercício — soma de um trio Pitagórico

```
def main():
  n = int(input("Digite um inteiro positivo: "))
  a = 1
  achei = False
  while not achei and a < n:
    b = a +1
    while not achei and a + b < n:
      c = n -a -b
      achei = pitagorico(a, b, c)
      if not achei:
        b += 1

      a += 1

  if achei:
    print("O número {} é a soma do trio pitagórico {}, {} e {}".format(n, a, b, c))
  else:
    print("O número {} não é a soma de nenhum trio pitagórico".format(n))
```

Exercício — soma de um trio Pitagórico

```
def main():
  n = int(input("Digite um inteiro positivo: "))
  a = 1
  achei = False
  while not achei and a < n:
    b = a +1
    while not achei and a + b < n:
      c = n -a -b
      achei = pitagorico(a, b, c)
      if not achei:
        b += 1
    if not achei:
      a += 1

  if achei:
    print("O número {} é a soma do trio pitagórico {}, {} e {}".format(n, a, b, c))
  else:
    print("O número {} não é a soma de nenhum trio pitagórico".format(n))
```

Exercício — calculando π

Dado um inteiro $k \geq 0$, é possível calcular um valor aproximado de π através da expansão de Gregory-Leibniz:

$$\pi \approx \frac{4}{1} - \frac{4}{3} + \frac{4}{5} - \frac{4}{7} + \frac{4}{9} - \frac{4}{11} \cdots + \frac{(-1)^k 4}{2k+1}$$

Escreva um programa que lê um valor para k e realiza o cálculo acima.

Exercício — calculando π

$$\pi \approx \frac{4}{1} - \frac{4}{3} + \frac{4}{5} - \frac{4}{7} + \frac{4}{9} - \frac{4}{11} \cdots + \frac{(-1)^k 4}{2k+1}$$

Exercício — calculando π

$$\pi \approx \frac{4}{1} - \frac{4}{3} + \frac{4}{5} - \frac{4}{7} + \frac{4}{9} - \frac{4}{11} \cdots + \frac{(-1)^k 4}{2k+1}$$

```
def main():
```

Exercício — calculando π

$$\pi \approx \frac{4}{1} - \frac{4}{3} + \frac{4}{5} - \frac{4}{7} + \frac{4}{9} - \frac{4}{11} \cdots + \frac{(-1)^k 4}{2k+1}$$

```
def main():
  k = int(input("Digite o valor de k: "))
```

Exercício — calculando π

$$\pi \approx \frac{4}{1} - \frac{4}{3} + \frac{4}{5} - \frac{4}{7} + \frac{4}{9} - \frac{4}{11} \cdots + \frac{(-1)^k 4}{2k+1}$$

```
def main():
  k = int(input("Digite o valor de k: "))









  print("O valor de pi é aproximadamente {}".format(pi))
```

Exercício — calculando π

$$\pi \approx \frac{4}{1} - \frac{4}{3} + \frac{4}{5} - \frac{4}{7} + \frac{4}{9} - \frac{4}{11} \cdots + \frac{(-1)^k 4}{2k+1}$$

```
def main():
  k = int(input("Digite o valor de k: "))



  while n <= k:



  print("O valor de pi é aproximadamente {}".format(pi))
```

Exercício — calculando π

$$\pi \approx \frac{4}{1} - \frac{4}{3} + \frac{4}{5} - \frac{4}{7} + \frac{4}{9} - \frac{4}{11} \cdots + \frac{(-1)^k 4}{2k+1}$$

```
def main():
  k = int(input("Digite o valor de k: "))


  n = 0
  while n <= k:



  print("O valor de pi é aproximadamente {}".format(pi))
```

Exercício — calculando π

$$\pi \approx \frac{4}{1} - \frac{4}{3} + \frac{4}{5} - \frac{4}{7} + \frac{4}{9} - \frac{4}{11} \cdots + \frac{(-1)^k 4}{2k+1}$$

```
def main():
  k = int(input("Digite o valor de k: "))


  n = 0
  while n <= k:


    n += 1
  print("O valor de pi é aproximadamente {}".format(pi))
```

Exercício — calculando π

$$\pi \approx \frac{4}{1} - \frac{4}{3} + \frac{4}{5} - \frac{4}{7} + \frac{4}{9} - \frac{4}{11} \cdots + \frac{(-1)^k 4}{2k+1}$$

```
def main():
  k = int(input("Digite o valor de k: "))
  pi = 0

  n = 0
  while n <= k:



    n += 1
  print("O valor de pi é aproximadamente {}".format(pi))
```

Exercício — calculando π

$$\pi \approx \frac{4}{1} - \frac{4}{3} + \frac{4}{5} - \frac{4}{7} + \frac{4}{9} - \frac{4}{11} \cdots + \frac{(-1)^k 4}{2k+1}$$

```
def main():
  k = int(input("Digite o valor de k: "))
  pi = 0

  n = 0
  while n <= k:
    pi +=          4 / (2*n +1)

    n += 1
  print("O valor de pi é aproximadamente {}".format(pi))
```

Exercício — calculando π

$$\pi \approx \frac{4}{1} - \frac{4}{3} + \frac{4}{5} - \frac{4}{7} + \frac{4}{9} - \frac{4}{11} \cdots + \frac{(-1)^k 4}{2k+1}$$

```
def main():
  k = int(input("Digite o valor de k: "))
  pi = 0

  n = 0
  while n <= k:
    pi += sinal * 4 / (2*n +1)

    n += 1
  print("O valor de pi é aproximadamente {}".format(pi))
```

Exercício — calculando π

$$\pi \approx \frac{4}{1} - \frac{4}{3} + \frac{4}{5} - \frac{4}{7} + \frac{4}{9} - \frac{4}{11} \cdots + \frac{(-1)^k 4}{2k+1}$$

```
def main():
  k = int(input("Digite o valor de k: "))
  pi = 0
  sinal = 1
  n = 0
  while n <= k:
    pi += sinal * 4 / (2*n +1)
    sinal *= -1
    n += 1
  print("O valor de pi é aproximadamente {}".format(pi))
```