MAC 110 — Introdução à Ciência da Computação

Aula 3

Nelson Lago

BMAC – 2024

Previously on MAC 110...

Programando

Programar envolve

1. **Compreender um problema em termos computacionais**
2. **Definir como esse problema pode ser solucionado (*algoritmo*)**
   - O algoritmo é *abstrato* (como a planta de um prédio ou uma receita de bolo)
3. ***Implementar* o algoritmo em uma linguagem de programação**
   - Gerando um *programa* que pode ser *executado* para solucionar o problema
4. **Testar o programa**

Programando

Para ser útil, um programa geralmente

1. **Obtém dados**
2. **"Faz alguma coisa" com esses dados**
   - Gerando um resultado
3. **"Faz alguma coisa" com esse resultado**
   - Mostra para o usuário
   - **Utiliza como dado para fazer outra coisa**

Expressões em python

- **A maioria das coisas em python são *expressões***
  - (expressões são coisas que têm um *valor*)
  - Exemplo: 47
  - Exemplo: $2 + 3$
  - Exemplo: "Oi galera!"
- **Expressões podem ser combinadas ou utilizadas como partes de outras expressões**
  - Exemplo: $\frac{2+3+7}{6}$
  - Exemplo: $2 + 3 + 7 < 0$
  - Exemplo: $2 + 3 + 7 < 0$ **and** $5 > 4$

Tipos

```
print(type(2))
print(type(False))
print(type(2.0))
print(type("Olá"))
```

```
<class 'int'>
<class 'bool'>
<class 'float'>
<class 'str'>
```

Nomes (variáveis)

- *Nomes* **permitem que pensemos mais sobre o problema a ser resolvido e menos sobre as idiossincrasias do computador**
- **Nomes em geral representam valores que *variam* (basicamente, alguma informação "real" que está em algum lugar na memória do computador)**
  - Como na matemática!
- **Por isso, chamamos esses nomes de "variáveis"**

Nomes (variáveis)

x ← 5 (atribuição)

Há um número finito de caracteres no teclado, então fazemos atribuição em python com “=”:

```
x = 5
x = "Olá, galera!"
x = y
x = x + 1
```

Nomes (variáveis)

- **Por que o título destes slides é “Nomes (variáveis)”?**
  - Não há muita graça em escrever programas como os que vimos acima, em que os dados são sempre os mesmos
  - Um dos principais usos de nomes é representar valores que *variam* (basicamente, alguma informação “real” que está em algum lugar na memória do computador)
    - » *Como na matemática!*
  - Por isso, chamamos esses nomes de “variáveis”
  - Acima, definimos o valor dos nomes (variáveis) com comandos de atribuição (`a = -1`) fixos. Como fazer se queremos valores que não sejam fixos?
  - `input()`

Primeiro programa – bully

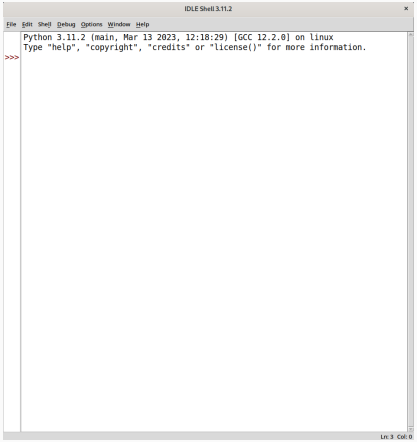
```
idade = input("Informe sua idade: ")
idade = int(idade)
print("Você tem só", idade, "anos?!?!")
print("Nossa, você aparenta ter", 2 * idade, "anos!")
```

and now for something completely different

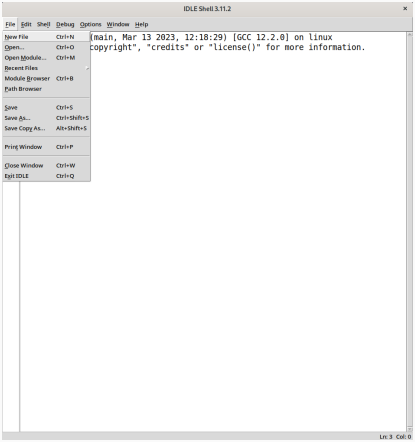
Adeus, shell!

- Começamos usando o *shell* do python através do IDLE
- MAS...
- Em geral, *não* se usa o shell; ele serve apenas para experimentar alguns comandos básicos
- O que fazemos é criar um arquivo de texto puro (mas usando um nome do tipo `.py` ao invés de `.txt`) com os comandos desejados
- Mas o que é "texto puro"?
- libreoffice vs IDLE

IDLE


IDLE Shell 3.11.2
File Edit Shell Debug Options Window Help
Python 3.11.2 (main, Mar 13 2023, 12:18:29) [GCC 12.2.0] on linux
Type "help", "copyright", "credits" or "license()" for more information.
>>>
Ln: 3 Col: 0

VS


IDLE Shell 3.11.2
File Edit Shell Debug Options Window Help
New File Ctrl+N
Open... Ctrl+O
Open Module... Ctrl+M
Recent Files
Module Browser Ctrl+B
Path Browser
Save Ctrl+S
Save As... Ctrl+Shift+S
Save Copy As... Alt+Shift+S
Print Window Ctrl+P
Close Window Ctrl+W
Exit IDLE Ctrl+Q
(main, Mar 13 2023, 12:18:29) [GCC 12.2.0] on linux
copyright", "credits" or "license()" for more information.
Ln: 3 Col: 0

Execução condicional 1/2

```
n = int(input("Digite um número natural: "))

if n % 2 == 0:
    print("O número", n, "é par!")
```

- **`n % 2 == 0`** → **`True`** ou **`False`**
  - Embora possamos ler "se condição", na verdade python faz "se o valor da expressão é verdadeiro (**`True`**)"
  - É **como se** python executasse **`if`** `condição == True`

Execução condicional 2/2

```
n = int(input("Digite um número natural: "))

if n % 2 == 0:
    print("O número", n, "é par!")

print("Ahazei!")
```

- **Ele sempre imprime “Ahazei!” ou só quando o número é par?**
- **Como ele sabe onde “acaba o efeito” do `if`?**
  - Qualquer quantidade de espaços, desde que seja consistente (4 espaços é o mais comum)

Execução condicional – `else`

```
n = int(input("Digite um número natural: "))
if n % 2 == 0:
    print("O número", n, "é par!")
else:
    print("O número", n, "é ímpar!")
print("Ahazei!")
```

- **A *indentação* indica os "lados" do condicional**
  - Sem variável ("n"), o programa sempre executaria o mesmo "lado"
- **O estado da variável só importa no momento do teste**
  - Se ela mudar em seguida, não afeta o condicional
- **Com `else`, os "lados" são mutuamente excludentes**
  - **Um** e **apenas um** deles é executado

Exercício — área e perímetro do quadrado 1/2

Dado o lado do quadrado, calcule a área e o perímetro

```
lado = int(input("Digite o lado do quadrado: "))
area = lado**2
perimetro = 4*lado
print("A área do quadrado é", area, "e o perímetro é", perimetro)
```

Exercício — área e perímetro do quadrado 2/2

Dado o lado do quadrado, calcule a área e o perímetro

```
lado = int(input("Digite o lado do quadrado: "))
print("A área do quadrado é", lado**2, "e o perímetro é", 4*lado)
```

Exercício — ordem crescente

Dados três números, verifique se eles estão em ordem crescente (considere que números iguais estão em ordem crescente)

```
a = int(input("Digite o primeiro número: "))
b = int(input("Digite o segundo número: "))
c = int(input("Digite o terceiro número: "))
if a <= b and b <= c:
    print("Os números estão em ordem crescente")
else:
    print("Os números não estão em ordem crescente")
```

Exercício — sistema de *login*

Crie um sistema que lê o UID (*login*) e a senha do usuário e, se os dados estiverem corretos, escreve “Bem-vindo!”; caso contrário, o sistema escreve “Login ou senha incorreto”. Os dados de acesso são “ali babá” e “abre-te, sésamo”.

```
uid = input("username: ")
senha = input("senha: ")
if uid == "ali babá" and senha == "abre-te, sésamo":
  print("Bem-vindo!")
else:
  print("Login ou senha incorreto")
```

Exercício — permissão de trabalho 1/2

Uma pessoa só pode trabalhar legalmente no Brasil se for maior de 18 anos ou se tiver entre 14 e 18 anos e for aprendiz. Além disso, só podem trabalhar os cidadãos brasileiros ou os estrangeiros com permissão de trabalho. Colete os dados relevantes do usuário e informe se ele pode ou não trabalhar.

```
brasileiro = input("Você é brasileiro? ")
permissão = input("Você tem permissão de trabalho no Brasil? ")
idade = int(input("Qual sua idade? "))
if brasileiro == "n" and permissão == "n" or idade < 14:
    print("Você não pode trabalhar no Brasil")
else:
    if idade < 18:
        print("Você pode trabalhar apenas como aprendiz")
    else:
        print("Você pode trabalhar no Brasil")
```

Exercício — permissão de trabalho 2/2

```
possível = True
if input("Você é brasileiro? ") == "n":
    if input("Você tem permissão de trabalho no Brasil? ") == "n":
        possível = False
if possível:
    idade = int(input("Qual sua idade? "))
    if idade < 14:
        possível = False
if not possível:
    print("Você não pode trabalhar no Brasil")
else:
    if idade < 18:
        print("Você pode trabalhar apenas como aprendiz")
    else:
        print("Você pode trabalhar no Brasil")
```

Repetições

Por que computação?

O computador é extremamente rápido, mas

- **É uma ferramenta com o mesmo nível de "inteligência" que um martelo**
  - Tudo tem que ser esmiuçado nos mínimos detalhes
    - » *"Vá à padaria e compre três pães"*
    - » *"Localize esta palavra no texto"*
    - » *...*

Não é mais fácil fazer manualmente?

Por que computação?

Às vezes, sim 🙂 mas:

- **Sistemas de controle**
- **Comunicação**
- **...**
- **Repetições**

Repetições “externas” e “internas”

- **Algumas repetições são externas ao programa**
  - Calculadora (o usuário faz inúmeros cálculos)
  - Jogo (cada partida é uma “repetição”)
  - ...
- **Mas, em geral, qualquer programa não-trivial vai realizar repetições internamente**
  - Xadrez (cada jogada é uma repetição)
  - Procurar uma palavra em um texto (várias comparações)
  - Apresentar uma foto na tela (cada pixel precisa ser “pintado” com a cor adequada)
  - ...

Tipos de repetição

- **Dois tipos fundamentais de repetição**
  1. Repetições até atingir um resultado
     - » *Encontrar o próximo primo*
     - » *Reiniciar o jogo até o usuário escolher "sair"*
     - » *...*
  2. Repetições sobre os elementos de um conjunto
     - » *Apresentar todos os pixels de uma foto na tela*
     - » *Trocar todas as letras de um texto para maiúsculas*
     - » *...*

Repetições e condições

- **Em ambos os casos, "algo" precisa acontecer para indicar que as repetições chegaram ao fim (o objetivo foi atingido ou todos os elementos do conjunto já foram processados)**

  (ok, às vezes queremos repetir indefinidamente, mas vamos ignorar isso por enquanto)

- **As repetições são controladas por algum tipo de *condição* baseada no estado de uma *variável***

```
chute = input("Adivinhe qual é minha cor favorita: ")
while chute != "rosa choque":
    chute = input("Errou! Tente novamente: ")
print("Acertou!")
```

Partes mínimas de um laço

- **Um laço correto precisa**
  - Inicializar a variável de controle antes do início do laço
  - Verificar a condição adequada a cada iteração para que as repetições aconteçam o número correto de vezes
  - Alterar o valor da variável de acordo com a lógica do programa (no mínimo, na última iteração) para garantir que o laço termine

Tipos de repetição

- **Dois tipos fundamentais de repetição**
  1. Repetições até atingir um resultado
     - » *Encontrar o próximo primo*
     - » *Reiniciar o jogo até o usuário escolher "sair"*
     - » *...*
  2. Repetições sobre os elementos de um conjunto
     - » *Apresentar todos os pixels de uma foto na tela*
     - » *Trocar todas as letras de um texto para maiúsculas*
     - » *...*

Repetições até atingir um resultado

```
chute = input("Adivinhe qual é minha cor favorita: ")
while chute != "rosa choque":
    chute = input("Errou! Tente novamente: ")
print("Acertou!")
```

- **A *condição* precisa mudar ao menos na última iteração!**
  - A *condição* testada é "a pessoa chutou `rosa choque`"
- **Sem variável, o programa nunca pararia de repetir**
  - (aqui, a variável é "`chute`")
    - » *(Onde está "* `== True` *"?)*

Partes mínimas de um laço

- **Um laço correto precisa**
  - Inicializar a variável de controle antes do início do laço
  - Verificar a condição adequada a cada iteração para que as repetições aconteçam o número correto de vezes
  - Alterar o valor da variável de acordo com a lógica do programa (no mínimo, na última iteração) para garantir que o laço termine

```
chute = input("Adivinhe qual é minha cor favorita: ")
while chute != "rosa choque":
    chute = input("Errou! Tente novamente: ")
print("Acertou!")
```

Repetições até atingir um resultado

- **Caso comum:**
  - branco sujo
  - amarelo icterícia
  - roxo hematoma
  - **rosa choque**

*A **variável** muda em todas as iterações, mas a **condição** só muda na última iteração*

- **Caso sortudo:**
  - **rosa choque**

***Nenhuma** iteração, então nem a variável nem a condição mudam*

- **Caso absurdo:**
  - roxo hematoma
  - roxo hematoma
  - roxo hematoma
  - **rosa choque**

*A variável **e** a condição só mudam na última iteração*

A variável sempre precisa mudar (ao menos) na última iteração!

Tipos de repetição

- **Dois tipos fundamentais de repetição**
  1. Repetições até atingir um resultado
     - » *Encontrar o próximo primo*
     - » *Reiniciar o jogo até o usuário escolher "sair"*
     - » *...*
  2. Repetições sobre os elementos de um conjunto
     - » *Apresentar todos os pixels de uma foto na tela*
     - » *Trocar todas as letras de um texto para maiúsculas*
     - » *...*

Repetições sobre os elementos de um conjunto

```
print("Prepare-se para o grito:")
n = 10
while n >= 0:
    print(n)
    n = n -1
print("AAAHHH!!!!")
```

- **O *conjunto* são os números 0–10 (às vezes a ordem importa!)**
- **A *condição* precisa mudar ao menos na última iteração!**
  - (indicando que todos os elementos do conjunto foram processados)
    - » *A condição testada é "*`n` *é maior ou igual a zero"*
- **Sem variável, o programa nunca pararia de repetir**
  - (aqui, a variável é "n")
    - » *(Onde está "* `== True` *"?)*

Partes mínimas de um laço

- **Um laço correto precisa**
  - Inicializar a variável de controle antes do início do laço
  - Verificar a condição adequada a cada iteração para que as repetições aconteçam o número correto de vezes
  - Alterar o valor da variável de acordo com a lógica do programa (no mínimo, na última iteração) para garantir que o laço termine

```
print("Prepare-se para o grito:")
n = 10
while n >= 0:
    print(n)
    n = n -1
print("AAAHHH!!!!")
```

Exercício — cálculo do fatorial 1/2

Cálculo do fatorial de um número (repetições sobre os elementos de um conjunto)

```
n = int(input("Digite um inteiro positivo: "))
fatorial = n
while (n > 2): # Número mágico
    n = n -1
    fatorial = fatorial * n
print(fatorial)
```

Exercício — cálculo do fatorial 2/2

Cálculo do fatorial de um número (usando o elemento neutro)

```
n = int(input("Digite um inteiro positivo: "))
fatorial = 1
while (n > 1): # Ou será >= 1 ?
    fatorial = fatorial * n
    n = n -1
print(fatorial)
```

É mais comum usar o valor da variável recebido
no início do laço e atualizar seu valor no final

(“principle of least surprise”)

Exercício — somatório 1/3

Dado um número inteiro positivo, calcular a soma dos n primeiros inteiros maiores que zero (repetições sobre os elementos de um conjunto)

```
n = int(input("Digite um inteiro positivo: "))
soma = n*(1+n) // 2
print("A soma dos", n, "primeiros inteiros é", soma)
```

Exercício — somatório 2/3

Dado um número inteiro positivo, calcular a soma dos n primeiros inteiros (usando um laço — com o elemento neutro)

```
n = int(input("Digite um inteiro positivo: "))
val = n
soma = 0
while val > 0:
    soma = soma + val
    val = val -1
print("A soma dos", n, "primeiros inteiros é", soma)
```

Exercício — somatório 2/3

Dado um número inteiro positivo, calcular a soma dos n primeiros inteiros (usando um laço — com um contador)

```
n = int(input("Digite um inteiro positivo: "))
soma = 0
i = 1
while i <= n:
    soma = soma + i
    i = i +1
print("A soma dos", n, "primeiros inteiros é", soma)
```

Exercício — somatório 1/2

Leia uma série de números terminada por zero fornecida pelo usuário e calcule sua soma (repetições até atingir um resultado)

```
n = int(input("Digite um número (zero para sair): "))
soma = 0
while n != 0:
    soma = soma + n
    n = int(input("Digite um número (zero para sair): "))
print("A soma dos números é " + str(soma))
```

Exercício — somatório 2/2

Leia uma série de números terminada por zero fornecida pelo usuário e calcule sua soma (repetições até atingir um resultado)

```
acabou = False
soma = 0
while not acabou:
    n = int(input("Digite um número (zero para sair): "))

    soma = soma + n
    if n == 0:
        acabou = True
print("A soma dos números é " + str(soma))
```

Exercício — contagem de dígitos

Dado um inteiro n e um dígito d ($0 \leq d \leq 9$), diga quantas vezes d aparece em n (repetições sobre os elementos de um conjunto).

```
n = int(input("Digite o número n: "))
d = int(input("Digite o dígito d: "))
val = n
conta = 0 # elemento neutro
while val > 0:
    este = val % 10
    if este == d:
        conta = conta +1
    val = val // 10
print("O dígito", d, "aparece", conta, "vezes em", n)
```