# MAC2166 - Introdução à Computação

**Prof. Dr. Helder Oliveira**

# Agenda

- Saída formatada
- Divisão inteira //
- Resto de divisão %

# Main

- O que é uma função main?
- A função main() é uma convenção em muitas linguagens de programação, mas em Python não é obrigatória. Seu objetivo é organizar o código em um ponto de entrada, melhorando a legibilidade e estrutura.

```
def main():
    print("Hello, World!")
```

- Não teremos nenhuma saída!

# Main

- Se o script for importado como módulo, a função main() **não será chamada** automaticamente.

```
def main():
    print("Executando main()")
main()
```

- Imprime a mensagem

# Saída formatada python

- Em Python é possível fazer impressões mais sofisticadas utilizando um formatador (%) seguido de um caractere indicando o tipo da variável ser impressa:
  - d ⇒ "inteiro"; f ⇒ "float"; c ⇒ "caractere"; s ⇒ "string".

```
m = 2
x = 2.3
t = 'A'

print("m=%d, x=%f, t=%c" % (m, x, t))
```

**Saída:**

```
m=2, x=2.300000, t=A
```

# Saída formatada python

- **Como funciona?**
- A string dentro do print usa **formatação com o operador %**.
- Isso significa que os valores das variáveis serão **inseridos na string no local dos marcadores de formato (%)**.
  - **%d**: É usado para formatar **números inteiros**. Ele substitui a variável com um número inteiro.
  - **%f**: É usado para formatar **números de ponto flutuante (decimais)**. Ele substitui a variável com um número decimal.
  - **%c**: É usado para formatar **caracteres** (ou seja, uma string de comprimento 1). Ele substitui a variável com um único caractere.

# Exemplos

- Imprimindo um número inteiro

```
num = 25
print("O número é: %d" % num)
```

- Imprimindo um número de ponto flutuante

```
pi = 3.14159
print("O valor de pi é: %.2f" % pi)
```

- Imprimindo uma string

```
nome = "João"
print("Olá, %s!" % nome)
```

- Imprimindo um número inteiro com sinal

```
saldo = -500
print("Seu saldo é: %+d" % saldo)
```

# Saída Formatada

Imprima:

```
| a:     1.34 | b:   342.02 |
| c:  3000.80 | d: 20900.98 |
```

**Solução:**

```
1 | print("| a: %8.2f | b: %8.2f |"%(a,b))
2 | print("| c: %8.2f | d: %8.2f |"%(c,d))
```

# Saída Formatada

- Qual a saída?

```
1  num_real1 = 13.2278430
2  num_real2 =  4.9
3  num_int1 = 121
4  num_int2 = 6
5  num_int3 = -57
6  print("Números reais formatados: |%10.3f|, |%10.3f|" %(num_real1, num_real2))
7  print("Números inteiros formatados: |%4d|, |%4d|, |%4d|" %(num_int1, num_int2, num_int3))
```

# Saída Formatada

- Qual a saída?

```
1  num_real1 = 13.2278430
2  num_real2 =  4.9
3  num_int1 = 121
4  num_int2 = 6
5  num_int3 = -57
6  print("Números reais formatados: |%10.3f|, |%10.3f|" %(num_real1, num_real2))
7  print("Números inteiros formatados: |%4d|, |%4d|, |%4d|" %(num_int1, num_int2, num_int3))
```

Saída do programa acima:

```
Números reais formatados: |    13.228|, |     4.900|
Números inteiros formatados: | 121|, |   6|, | -57|
```

# Exemplo

- Imprimindo múltiplas variáveis

```
def main():
    nome = "Maria"
    idade = 30
    altura = 1.75
    print("Nome: %s, Idade: %d, Altura: %.2f" % (nome, idade, altura))

# Chama a função main diretamente
main()
```

- Saída:

```
Nome: Maria, Idade: 30, Altura: 1.75
```

# Exercício

- Crie uma variável numero com o valor inteiro 42. Imprima essa variável utilizando o formatador %d, que é usado para imprimir números inteiros.

# Exercício

- Crie uma variável numero com o valor inteiro 42. Imprima essa variável utilizando o formatador %d, que é usado para imprimir números inteiros.

```
numero = 42
print("Número: %d" % numero)
```

# Exercício

- Crie uma variável valor com o valor 3.14159. Imprima essa variável utilizando o formatador %.1f, para mostrar o número com 1 casa decimal.

# Exercício

- Crie uma variável valor com o valor 3.14159. Imprima essa variável utilizando o formatador %.1f, para mostrar o número com 1 casa decimal.

```
valor = 3.14159
print("Valor: %.1f" % valor)
```

# Exercício

- Crie uma variável nome com o valor "Carlos". Imprima essa variável utilizando o formatador %s, que é usado para imprimir strings.

```
nome = "Carlos"
print("Nome: %s" % nome)
```

# Exercício

- Crie três variáveis:
  - cidade: com o valor "São Paulo"
  - populacao: com o valor 12300000
  - area: com o valor 1521.25
- Imprima todas essas variáveis em uma única linha utilizando o formatador %s para strings, %d para inteiros e %.2f para números de ponto flutuante com 2 casas decimais.

# Exercício

- Crie três variáveis:
  - cidade: com o valor "São Paulo"
  - populacao: com o valor 12300000
  - area: com o valor 1521.25
- Imprima todas essas variáveis em uma única linha utilizando o formatador %s para strings, %d para inteiros e %.2f para números de ponto flutuante com 2 casas decimais.

```
cidade = "São Paulo"
populacao = 12300000
area = 1521.25
print("Cidade: %s, População: %d, Área: %.2f" % (cidade, populacao, area))
```

# Exercício

- Crie uma variável divida com o valor -1500. Imprima essa variável utilizando o formatador %+d, que exibe o sinal (+ ou -) junto ao número.

# Exercício

- Crie uma variável divida com o valor -1500. Imprima essa variável utilizando o formatador %+d, que exibe o sinal (+ ou -) junto ao número.

```
divida = -1500
print("Dívida: %+d" % divida)
```

# Exercício

- Crie uma variável numero com o valor 45. Imprima essa variável com uma largura de campo de 6 caracteres, utilizando o formatador %6d. Isso significa que o número será impresso com 6 caracteres de largura, e espaços à esquerda serão adicionados, se necessário.

# Exercício

- Crie uma variável numero com o valor 45. Imprima essa variável com uma largura de campo de 6 caracteres, utilizando o formatador %6d. Isso significa que o número será impresso com 6 caracteres de largura, e espaços à esquerda serão adicionados, se necessário.

```
numero = 45
print("Número com largura 6: %6d" % numero)
```

# Exercício

- Crie uma variável codigo com o valor 123. Imprima essa variável utilizando o formatador %05d para que o número seja exibido com 5 dígitos, preenchendo os espaços à esquerda com zeros.

# Exercício

- Crie uma variável código com o valor 123. Imprima essa variável utilizando o formatador %05d para que o número seja exibido com 5 dígitos, preenchendo os espaços à esquerda com zeros.

```
codigo = 123
print("Código com zeros à esquerda: %05d" % codigo)
```

# Saída formatada python

- Para usar strings literais formatadas, comece uma string com f ou F, antes de abrir as aspas ou aspas triplas.
- Dentro dessa string, pode-se escrever uma expressão Python entre caracteres { e }, que podem se referir a variáveis, ou valores literais.

```
>>> ano = 2016
>>> evento = 'Referendo'
>>> f'Resultados do {evento} {ano}'
'Resultados do Referendo 2016'
```

# Saída formatada python

- O método de strings str.format() requer mais esforço manual.
  - 42_572_654 é uma forma de representar 42.572.654, melhorar a legibilidade.
- Dois exemplos de como formatar variáveis:

```
>>> votos_sim = 42_572_654
>>> votos_totais = 85_705_149
>>> porcentagem = votos_sim / votos_totais
>>> '{:-9} votos SIM  {:2.2%}'.format(votos_sim, porcentagem)
' 42572654 votos SIM  49.67%'
```

  - **'{:-9}'**: O número fica com 9 espaços).
  - **'{:2.2%}'**: Isso converte o número da porcentagem para um formato de porcentagem. A parte 2.2% significa que ele vai mostrar a porcentagem com **duas casas decimais** e com o símbolo %.

# Saída formatada python

- **'{:-9}'**:
  - **:** O : após as chaves é usado para **iniciar o especificador de formatação**.
  - **-**: Esse sinal indica que queremos **alinhamento à esquerda**. Ou seja, o número será alinhado à esquerda dentro do espaço reservado.
  - **9**: O número 9 representa a **largura mínima** do campo. Ou seja, a string gerada terá **pelo menos 9 caracteres de comprimento**. Se o valor que estamos formatando for menor que 9 caracteres, o Python adicionará **espaços** à direita para preencher até 9 caracteres. Caso o valor seja maior que 9 caracteres, ele ocupará todo o espaço, sem truncar ou cortar nada.

# Saída formatada python

- Como funciona:

```
numero = 42572654
formatted = '{:-9}'.format(numero)
print(formatted)
```

- Saída

```
42572654
```

- Note que o número tem **9 espaços reservados**, mas como ele tem apenas **8 dígitos**, o Python preenche o restante com espaços **à direita** (alinhamento à esquerda).

# Saída formatada python

- **{:2.2%}**
  - **:**: Novamente, o : serve para **iniciar a formatação**.
  - **2.2**: Esses dois números indicam a **precisão** da formatação. O número 2 antes do ponto refere-se à **largura mínima do campo**, enquanto o 2 depois do ponto indica a **quantidade de casas decimais**.
  - **%**: Esse símbolo indica que estamos formatando um número **como uma porcentagem**.

# Saída formatada python

- Como funciona?

```
porcentagem = 0.4967
formatted = '{:2.2%}'.format(porcentagem)
print(formatted)
```

- Saída

```
49.67%
```

- **2 (largura mínima do campo)**: A largura mínima do campo onde o valor será exibido. Ou seja, o valor formatado ocupará pelo menos **2 espaços** na string.
- **.2 (número de casas decimais)**: O número após o ponto determina a **quantidade de casas decimais** que o número exibido terá. Aqui, o número será arredondado para **2 casas decimais**.
- **% (percentual)**: Ao usar o símbolo %, o número será **convertido para porcentagem**, multiplicando o valor por 100 e exibindo o símbolo % ao final.

# O método format()

- Um uso básico do método **str.format()** tem esta forma:

```
>>> print('Nós somos os {} que dizem "{}!"'.format('cavaleiros', 'Ni'))
Nós somos os cavaleiros que dizem "Ni!"
```

- As chaves e seus conteúdos (chamados campos de formatação) são substituídos pelos objetos passados.

```
>>> print('{0} e {1}'.format('spam', 'ovos'))
spam e eggs
>>> print('{1} e {0}'.format('spam', 'ovos'))
eggs e spam
```

# O método format()

- Se argumentos nomeados são passados para o método str.format(), seus valores serão referenciados usando o nome do argumento:

```
>>> print('Este {comida} está {adjetivo}.'.format(
...       comida='spam', adjetivo='absolutamente horrível'))
Este spam está absolutamente horrível.
```

- Argumentos posicionais e nomeados podem ser combinados à vontade:

```
>>> print('A história de {0}, {1} e {outro}.'.format('Bill', 'Manfred',
...                                                  outro='Georg'))
A história de Bill, Manfred e Georg.
```

# Modificador de formato !a

- Esse modificador aplica função de **conversão** nos valores que você está formatando.
- **!a:** Aplica a função **ascii()** que converte um valor para uma string de **representação ASCII**.
- Ela exibe a versão de um valor em ASCII, tratando os caracteres especiais de forma mais visível (por exemplo, convertendo os caracteres não ASCII em escape codes).

```
valor = "Olá, é um prazer!"
formatted = '{!a}'.format(valor)
print(formatted)
```

**Saída:**

```
'Ol\u00e1, \u00e9 um prazer!'
```

# Modificador de formato !s

- A função **str()** converte o valor para uma **string normal** (o comportamento padrão de conversão de um valor para texto).
- Usando **!s**, o Python vai chamar a função str() no valor antes de formatá-lo.

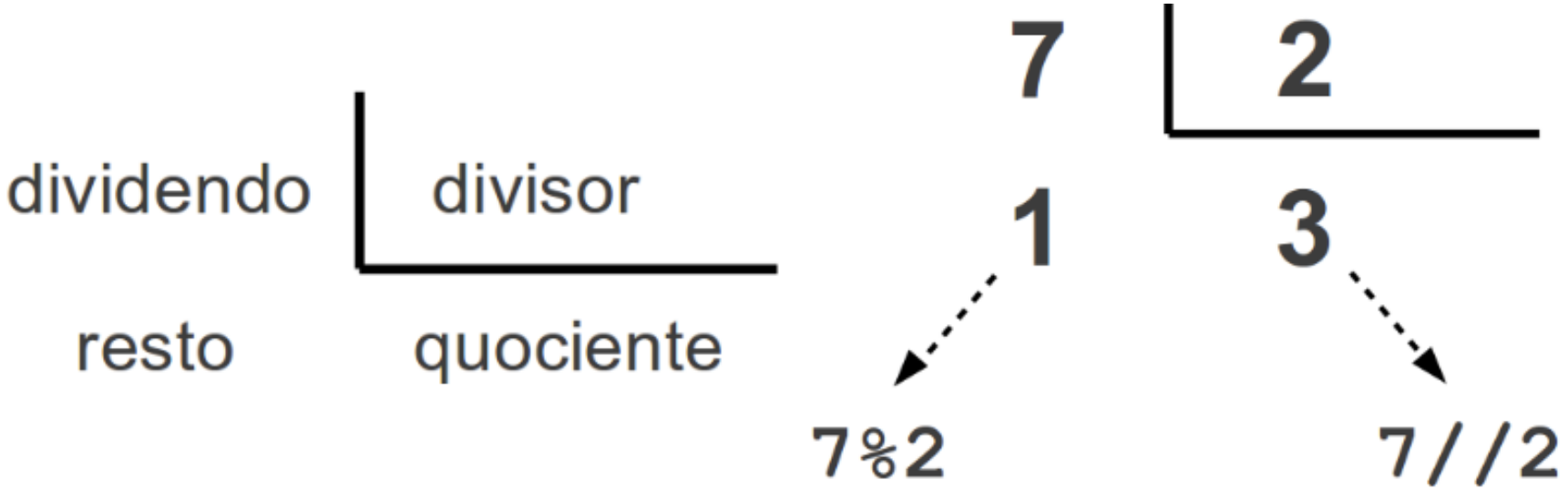
```
valor = 123
formatted = '{!s}'.format(valor)
print(formatted)
```

**Saída:**

```
123
```

- **!s** converte o valor 123 para sua **representação em string**. No caso de números inteiros, o resultado da conversão será exatamente o número, mas se fosse um número decimal, a conversão para string manteria o valor no formato decimal.

# Operadores Matemáticos



# Operadores Matemáticos - Divisão

- Divisão:

```
7 / 2
# 3.5
```

```
a = 10
a / 7
# 1.4285714285714286
```

- Divisão Inteira:

```
7 // 2
# 3
```

```
a = 10
a // 3.4
# 2.0
```

# Exercício

- Solicite ao usuário que insira dois números inteiros e imprima o resultado da divisão inteira entre esses dois números.

# Exercício

- Solicite ao usuário que insira dois números inteiros e imprima o resultado da divisão inteira entre esses dois números.

```
# Solicita os números ao usuário
num1 = int(input("Digite o primeiro número: "))
num2 = int(input("Digite o segundo número: "))
# Imprime o resultado da divisão inteira
print("Resultado da divisão inteira:", num1 // num2)
```

# Operadores Matemáticos - Módulo

- Módulo: resto da divisão inteira.

```
57 % 13
# 5
```

```
3 % 2
# 1
```

```
5.5 % 2
# 1.5
```

```
5 % 1.5
# 0.5
```

# Exercício

- Solicite ao usuário que insira dois números inteiros e imprima o resto da divisão entre eles.

# Exercício

- Solicite ao usuário que insira dois números inteiros e imprima o resto da divisão entre eles.

```
# Solicita os números ao usuário
num1 = int(input("Digite o primeiro número: "))
num2 = int(input("Digite o segundo número: "))
# Imprime o resto da divisão
print("Resto da divisão:", num1 % num2)
```

# Exercício

- Solicite ao usuário que insira um número e imprima se ele é par ou ímpar, usando o operador %.

# Exercício

- Solicite ao usuário que insira um número e imprima se ele é par ou ímpar, usando o operador %.

```
# Solicita o número ao usuário
numero = int(input("Digite um número: "))
# Imprime "Par" ou "Ímpar" com base no resto da divisão por 2
print("Par" * (numero % 2 == 0) + "Ímpar" * (numero % 2 != 0))
```

# Atualizações Compactas

- x %= y  é equivalente a x = x % y.
- x = int(x / y) é equivalente a x = x // y

# Erros Comuns com Operadores Matemáticos

- Divisão por zero:

```
10 / 0
# ZeroDivisionError: division by zero
```

```
10 / 0.0
# ZeroDivisionError: float division by zero
```

```
2 // 0
# ZeroDivisionError: integer division or modulo by zero
```

```
2 // 0.0
# ZeroDivisionError: float divmod()
```

```
10 % 0
# ZeroDivisionError: integer division or modulo by zero
```

```
10 % 0.0
# ZeroDivisionError: float modulo
```

# Exercício

- Faça um programa que receba dois números e imprima **True** se o primeiro for múltiplo do segundo, e **False** caso contrário.
  - Obs: utilize o operador %

# Exercício

- Faça um programa que receba dois números e imprima **True** se o primeiro for múltiplo do segundo, e **False** caso contrário.

```
a = int(input("Digite o primeiro número: "))
b = int(input("Digite o segundo número: "))
# Verifica se o primeiro número é múltiplo do segundo
print(a %b == 0)
```

# Exercício

- Faça um programa que receba um número e retorne **True** se ele for par e **False** caso contrário.
  - Obs: utilize o operador %

# Exercício

- Faça um programa que receba um número e retorne **True** se ele for par e **False** caso contrário.

```
# Recebe o número do usuário
numero = float(input("Digite um número: "))

# Verifica se o número é par (sem usar // ou %)
resultado = (numero - 2 * int(numero / 2)) == 0

# Exibe o resultado
print(resultado)
```

```
# Recebe o número do usuário
numero = float(input("Digite um número: "))

# Verifica se o número é par
print(numero % 2 == 0)
```

# Exercício

- Solicite um número e retorne **True** se ele for positivo **e** par ao mesmo tempo.
  - Obs: utilize o operador %

# Exercício

- Solicite um número e retorne True se ele for positivo **e** par ao mesmo tempo.

```
# Recebe o número do usuário
numero = float(input("Digite um número: "))
# Verifica se o número é positivo e par ao mesmo tempo
resultado = (numero > 0) * ((numero - 2 * int(numero / 2)) == 0)
print(bool(resultado))
```

```
# Recebe o número do usuário
numero = float(input("Digite um número: "))
# Verifica se o número é positivo e par ao mesmo tempo
print(numero > 0 and numero % 2 == 0)
```

# Exercício

- Peça um número e verifique se ele é múltiplo de 3 **e** múltiplo de 5 ao mesmo tempo.
  - Obs: utilize o operador %

# Exercício

- Peça um número e verifique se ele é múltiplo de $3$ **e** múltiplo de $5$ ao mesmo tempo.

```
# Recebe o número do usuário
numero = float(input("Digite um número: "))

# Verifica se o número é múltiplo de 3 e 5 ao mesmo tempo
resultado = (numero / 3 == int(numero / 3)) * (numero / 5 == int(numero / 5))

# Exibe o resultado
print(bool(resultado))
```

```
# Recebe o número do usuário
numero = float(input("Digite um número: "))

# Verifica se o número é múltiplo de 3 e 5 ao mesmo tempo
print(numero % 3 == 0 and numero % 5 == 0)
```

# Exercício

- Crie duas variáveis:
  - a = 15
  - b = 4
- Imprima:
  1. O resultado da **divisão inteira** de a por b usando o operador //.
  2. O **resto da divisão** de a por b usando o operador %.

# Exercício

- Crie as variáveis:
  - num1 = 45
  - num2 = 7.5
- Imprima a divisão de num1 por num2 com:
  1. Apenas 2 casas decimais utilizando a saída formatada %.2f.
  2. O resultado da divisão inteira entre num1 e num2, utilizando o operador //.

# Exercício

- Crie um programa que faça a divisão inteira e o cálculo do resto de divisão para os números de 1 até 10 quando divididos por 3. Imprima o resultado formatado de cada iteração no seguinte formato:
  - Número: x | Divisão Inteira: y | Resto da Divisão: z

# Exercício

- Crie uma função chamada dividir que recebe dois parâmetros x e y. A função deve:
  1. Retornar a **divisão inteira** de x por y.
  2. Retornar o **resto** da divisão de x por y.
  3. Exibir ambos os resultados de forma formatada.

# Exercício

- Você tem as variáveis:
  - preco = 152.75 (preço de um produto)
  - quantidade = 3 (quantidade comprada)
- Imprima o valor total a ser pago, com 2 casas decimais, no seguinte formato:
  - Total a ser pago: R$ xxx.xx

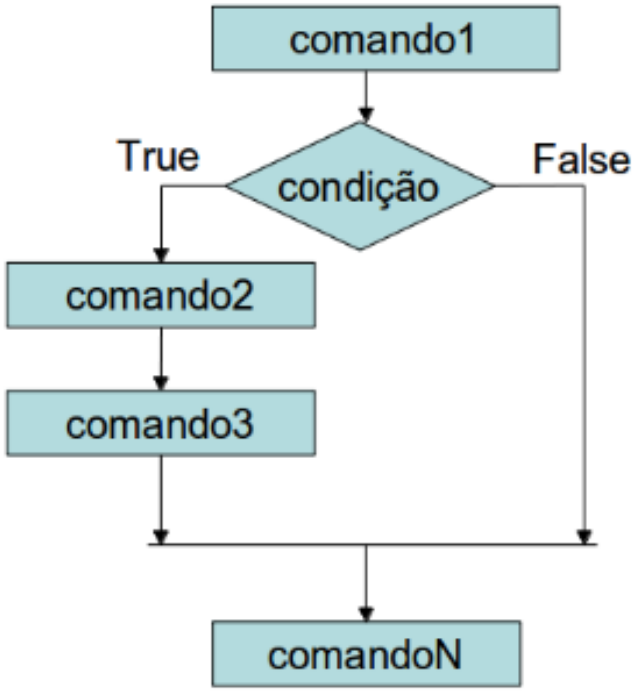
# Exercício

- Crie uma tabela de multiplicação para o número 7. Para cada número de 1 a 10, imprima a multiplicação de 7 por esse número com o seguinte formato:
  - 7 x 1 = 7
  - 7 x 2 = 14

# Exercício

- Agora, crie uma tabela de multiplicação para os números de1 até 9. Para cada número de 1 a 10, imprima a multiplicação do numero por esse número com o seguinte formato:
  - 1 x 1 = 1
  - 1 x 2 = 2

# Execução condicional




```
comando1
if condição :
    # bloco de comandos.
    comando2
    comando3
    :

comandoN
```

# Exercicio

- Dados um número inteiro n, n>0, e uma sequência com n números inteiros, determinar quantos números da sequência são pares.
- Para 6 -2 7 0 -5 8 4.O seu programa deve escrever o número 5 para o número de pares.

# Exercicio

- Dados um número inteiro n, n>0, e uma sequência com n números inteiros, determinar quantos números da sequência são pares.
- Para 6 -2 7 0 -5 8 4. O seu programa deve escrever o número 5 para o número de pares.

```
1  n = int(input("Digite o tam da seq: "))
2  conta_par = 0 # Contador de números pares encontrados.
3  i = 1
4  while i <= n:
5      num = int(input("Digite um num da seq: "))
6      if num % 2 == 0: # Testa se num é par.
7          conta_par = conta_par + 1
8      i = i + 1
9  print("Quant. pares =", conta_par)
10 # Fim do programa.
```

# Dúvidas

# Referencias

- Slides baseados em:
  - Zanoni Dias, MC102- Algoritmos e Programação de Computadores, 2020.