COMPUTAÇÃO 1 – AULA 2

Tipos de dados e Variáveis

Prof. Cesar Raitz

1. Introdução

- Um **bit** é a menor unidade de informação!
- Na memória dos computadores, os dados são armazenados em **conjuntos de bits**.
- Você pode referenciar dados (número etc.) usando **variáveis**.
- Mas como sequências de 0 e 1 podem representar músicas, fotos e vídeos?

2. Do binário ao significado

- Para saber o que uma sequência de bits significa, precisamos saber *como interpretar os bits*.
- Quantos **inteiros** podemos armazenar com 4 bits?

Binário	Decimal
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7

Binário	Decimal
1000	8
1001	9
1010	10
1011	11
1100	12
1101	13
1110	14
1111	15

- Um grupo de 8 bits forma um **byte** que representa $2^8 = 256$ valores distintos.
- Uma forma de representar **inteiros negativos** é usar o primeiro bit à esquerda como sinal. Por exemplo (agora 5 bits):

Binário	Decimal
0.0011	3
0.1010	10
0.1111	15

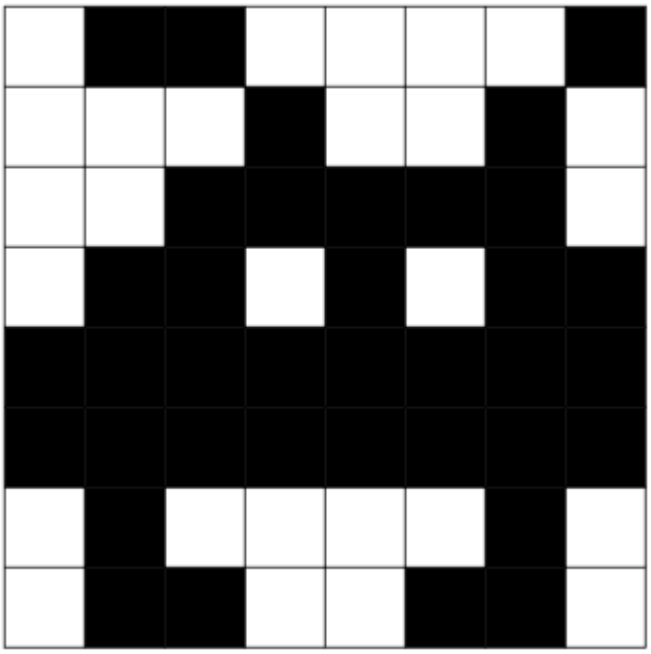
Binário	Decimal
1.0011	-3
1.1010	-10
1.1111	-15

- Agora, textos são compostos de caracteres (letras, números, pontuação *etc.*)
- A forma mais comum de representar caracteres é a tabela ASCII.
  Veja alguns exemplos:

Binário	Decimal	Caractere
0100 0001	65	A
0100 0010	66	B
0100 0011	67	C
	...	
0101 1001	89	Y
0101 1010	90	Z

Binário	Decimal	Caractere
0110 0001	97	a
0110 0010	98	b
0110 0011	99	c
	...	
0111 1001	121	y
0111 1010	122	z

- Um bit pode determinar quando um pixel fica aceso ou apagado na tela do computador.
- Suponha que 0 = aceso e 1 = apagado.
  Então, o mesmo número que representa o caractere ‘a’ (0110 0001) pode ser usado para desenhar a primeira linha da figurinha abaixo:



- As imagens coloridas usam o sistema RGB (*Red, Green, Blue*), onde cada pixel é uma mistura de 3 cores, com intensidades que variam entre 0 e 255.
- Para representar caracteres da nossa língua, como ç, ã, á, ... Precisamos usar a representação UTF-8, mais moderna.

3. Variáveis

- Já vimos que **`print()`** mostra coisas na tela (Terminal).
- Será que podemos guardar um valor na memória para mostrá-lo depois? Mas é claro!

```
1. x = 5
2. print(x)
```

- Na linha 1, foi **criada uma variável** x com o valor 5 **atribuído** a ela.
- Na linha 2, o comando **`print(x)`** imprime o valor atual de `x`.

- Ao rodar o programa, o resultado no Terminal será:

```
5
```

- *Sempre que o Python encontra o nome de uma variável, a substitui imediatamente pelo seu valor.*
- Memorize a sintaxe para **definição/criação de variáveis**:

```
<nome da variável> = <valor atribuído>
```

- Você também pode criar variáveis que já recebem o resultado de alguma conta:

```
1. x = 10 // 3
2. y = 10 % 3
3. print(x)
4. print(y)
```

- Também é possível usar variáveis para fazer contas:

```
1. x = 5
2. y = 3
3. print(x+y)
4. print(y-x)
5. print(x*y)
6. print(y/x)
```

- Qual será o resultado do seguinte programa?

```
1. x = 10 // 3
2. y = 10 % 3
3. print(x)
4. print(y)
```

Nomes de variáveis?

- A seguinte lista fornece boas práticas para nomear variáveis:
  - Pode usar letras maiúsculas e minúsculas, *mas prefira usar minúsculas*. Também evite usar acentuação e cedilha:

    🙂 soma
    🙁 Divisao
    🙁 subtração
    🙁 MULTIPLICACAO

  - O único caractere especial que pode ser usado é o *underline*:

    ✅ soma_total
    ❌ soma total?
    ❌ soma-total

  - Pode usar números depois do primeiro caractere:

    ✅ juros1
    ✅ bairros35
    ❌ 7samurais

  - Nomes curtos, como `x` e `y`, são bons para contas rápidas, mas não para usar ao longo do programa. *Prefira nomes significativos* para o programa, não os enigmáticos:

    ✅ nome_aluno
    ✅ idade_aluno
    ❌ variavel35
    ❌ cleytinho

Exercício 1. Encontre os erros.

a)
```
1. valor_inicial = 100
2. taxa_juros = 20/100
3. juros_simples = valor_inicial * taxa_juros * meses
4. print(juros_simples)
5. meses = 4
```

b)
```
1. 7 = P1
2. 9.8 = P2
3. media = (P1 + P2) / 2
4. print("A média das provas é")
5. print(media)
```

c)
```
1. 1o_valor
2. 2o_valor = 1o_valor + 1
3. print 1o_valor
```

4. Tipos de variáveis

- O **tipo de variável** é definido pelo tipo de valor guardado:
  - **int** guarda números inteiros
  - **float** guarda números reais
  - **complex** guarda números complexos
  - **str** guarda textos ou caracteres, chamados de **strings**
  - **bool** guarda valores binários, representados por `True` e `False`

- Veja alguns exemplos:

```
1. idade = 32           # idade é um int
2. taxa = 0.25          # taxa é um float
3. raiz = 3 + 5j        # raiz é um complex
4. nome = "Adalberto"   # nome é uma str
5. servido = True       # servido é um bool
```

- Geralmente, o Python não deixa colocar mais de uma instrução por linha mas...
- Tudo que vem depois de # é ignorado! São **comentários**.
- Servem para esclarecer o código

5. Perguntando coisas

- Até agora, fizemos programas para calcular coisas, como a hipotenusa de um triângulo retângulo:

```
1. cateto1 = 3
2. cateto2 = 4
3. hipotenusa = cateto1**2 + cateto2**2
4. hipotenusa = hipotenusa**0.5
5. print(hipotenusa)
```

- Para cada triângulo retângulo, precisamos alterar os valores no programa 😟
- E se pudéssemos **perguntar** os valores?
- Para isso temos a função **`input()`**. Memorize a sintaxe:

```
<variável resposta> = input(<pergunta>)
```

- `<pergunta>` é a pergunta que se faz para quem está usando o programa, deve ser uma *string*.
- `<variável resposta>` é uma variável que deve guardar a resposta digitada pelo usuário, *também é uma string!*
- Por exemplo:

```
1. nome = input("Como você se chama? ")
2. print("Muito prazer ")
3. print(nome)
```

- Veja a saída do programa no Terminal:

```
Como você se chama? Juninho <ENTER>
Muito prazer
Juninho
```

- Note que a pergunta está entre aspas duplas, porque é uma *string*.
- **Uma string é um texto entre aspas duplas ou simples.**
- Para que a saída fique profissa, pega essa dica:
  Podemos somar duas strings 🤯

```
1. nome = input("Como você se chama? ")
2. print("Muito prazer " + nome + "!")
```

```
Como você se chama? Camila <ENTER>
Muito prazer Camila
```

Exemplo: Calculando a média das notas

```
1. p1 = input("Qual foi sua P1? ")
2. p2 = input("Qual foi sua P2? ")
3. media = (p1 + p2)/2
4. print("Sua média é " + media)
```

- Parece que vai funcionar, mas...

```
Qual foi a sua P1? 8.6 <ENTER>
Qual foi a sua P2? 10 <ENTER>
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: unsupported operand type(s) for /:
'str' and 'int'
```

- Na linha 3, temos a soma de duas strings, `p1+p2`, uma operação bem definida
- Mas depois, o programa tenta dividir a string `p1+p2` por pelo inteiro 2, *uma operação desconhecida pelo Python!*
- Qual é a solução? Para fazer contas, precisamos converter as strings `p1` e `p2` para **float**.

```
<valor em float> = float(<valor em string>)
```

```
1. p1 = input("Qual foi sua P1? ")
2. p2 = input("Qual foi sua P2? ")
3. p1 = float(p1)
4. p2 = float(p2)
5. media = (p1 + p2)/2
6. print("Sua média é " + str(media))
```

- Agora temos uma operação bem definida, a divisão do número **float** `(p1 + p2)` pelo **int** 2.

```
Qual foi a sua P1? 8.6 <ENTER>
Qual foi a sua P2? 10 <ENTER>
Sua média é 9.3
```

6. Mais exercícios

Exercício 2. Calculando juros e montantes.

Crie um programa para calcular juros simples e compostos, as fórmulas para os juros são:

$$J_s = C \times i \times m$$

$$J_c = C[(1 + j)^m - 1]$$

- J_s é os juros simples
- C é o capital inicial
- i é a taxa de juros simples mensal
- m é o número de meses da aplicação
- J_c é os juros compostos
- j é a taxa de juros compostos ao mês

O montante, em cada caso, é a soma do capital inicial mais os juros.