

# Variáveis, Tipos de Dados e Operadores em Python

Afonso Paiva  
apneto@icmc.usp.br

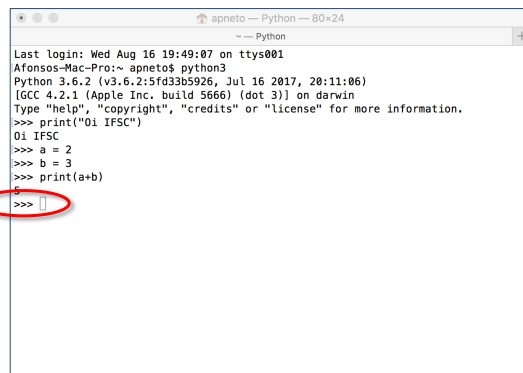
Fundamentos da Programação de Computadores – SME0332

1

## IDLE

(Integrated Development and Learning Environment)

- Python conta com uma interface interativa para execução de pequenas sequências de comandos
  - basta chamar **python3** no terminal (CMD ou prompt)



```
apneto ~ Python — 80x24
~ -- Python
Last login: Wed Aug 16 19:49:07 on ttys001
Afonso-Mac-Pro:~ apneto$ python3
Python 3.6.2 (v3.6.2:5fd33b5926, Jul 16 2017, 20:11:06)
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> print("01 IFSC")
01 IFSC
>>> a = 2
>>> b = 3
>>> print(a+b)
5
>>> []
```

o prompt **>>>** significa  
que estamos no IDLE

4

## Dados

- Um **dado** é uma informação que um algoritmo recebe e manipula.
- Exemplos de dados:
  - Nomes
  - Datas
  - Valores (notas, preços, altura, temperatura,...)
  - Condições (falso ou verdadeiro)

5

## Tipos de Dados

- O **tipo de um dado** define o conjunto de valores ao qual o valor do dado pertence, bem como o conjunto de todas as operações que podem atuar sobre qualquer valor daquele conjunto de valores.
- Os tipos de dados mais básicos em algoritmos são:
  - Caractere
  - Numérico
  - Lógico

6

## Tipos de Dados: Numérico

- **Inteiro (int):** representa um número inteiro;
  - Exemplos: -35, -20, 0, 7, 14, 34
  - Podem ser usados para idade em anos, número de filhos, etc...
- **Real (float):** representa um número real. Também é conhecido como *ponto flutuante*.
  - Exemplos: 3.1415, -234.46, 45.15
  - Podem ser usados para saldo bancário, altura, peso, temperatura, etc...

7

## Tipos de Dados: Caractere (char)

- Toda e qualquer informação composta por um conjunto de caracteres alfanuméricos:
  - numéricos (algarismos): '0','1',...,'9';
  - alfabéticos (letras): 'A','B',...,'Z','a','b',...,'z';
  - especiais (símbolos) '@', '\*', '#', '!', '\$', '?'...;
- Caracteres podem ser usados para a codificação de alguns itens, tais como:
  - sexo ('m', 'f');
  - estado civil ('s','c','d','v').

8

## Tipos de Dados: **String**

- String = conjunto de caracteres;
- Exemplos:
  - “nome”
  - “Brasil”
  - “1,80m”
  - “CEP”
  - “e-mail”

9

## Tipos de Dados: **Lógico**

- Dados lógicos podem assumir apenas dois valores: **verdadeiro** (True ou 1) ou **falso** (False ou 0).
- Também conhecido como *booleano* (**bool**).
- São usados para expressar uma condição:
  - $4 > 5$  é falso;
  - se o cheque número 000123 já foi compensado, ou não.

10

## Constantes

- Um dado que não sofre alteração no decorrer do tempo, ou seja, seu valor é **constante** desde o início até o fim da execução do algoritmo.
- Exemplos:
  - $\pi = 3.1416$ ;
  - `salario_minimo = 415.00`;
  - `CPF = 2222222222`;

11

## Variáveis

- Um dado é classificado variável quando tem a possibilidade de ser alterado em algum instante no decorrer do tempo.
- Durante a execução do algoritmo o valor do dado pode sofrer alteração.
- Exemplos: velocidade de um carro, taxa de juros, temperatura.

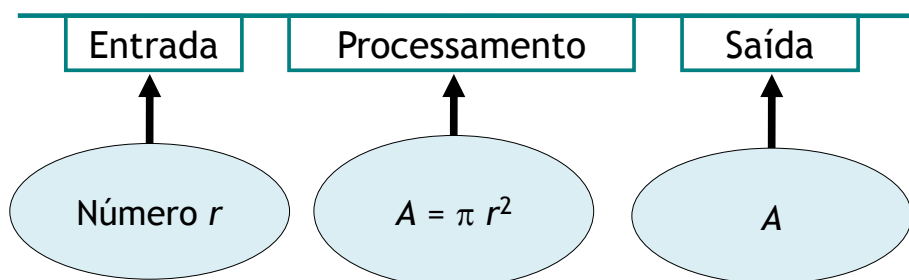
12

## Constantes X Variáveis

- Algoritmo para calcular a área de uma circunferência.
  - Fórmula da área:  $A = \pi r^2$
  - Constante ?
  - Variável ?
  - Que tipo de dado podemos definir a  $A$  ?

13

## Algoritmo: cálculo da área de um círculo



Onde as informações (dados de entrada, resultados parciais) são armazenados durante a execução do programa?

14

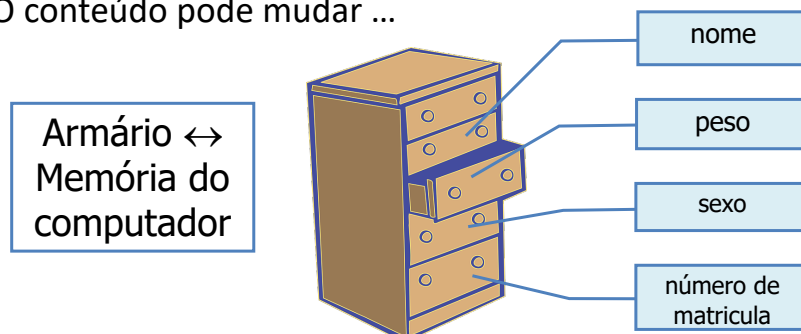
## Declaração de Variáveis

- Na **memória** do computador (hardware);
  - memória  $\leftrightarrow$  armário com gavetas;
- Cada gaveta possui um nome (**identificador**) e guarda uma informação (**conteúdo**);
- A gaveta tem uma localização física (**endereço**) responsável para armazenar o conteúdo.

15

## Variáveis

- As linguagens de programação permitem que os usuário atribuam nomes para as posições de memória da máquina.
- O conteúdo pode mudar ...



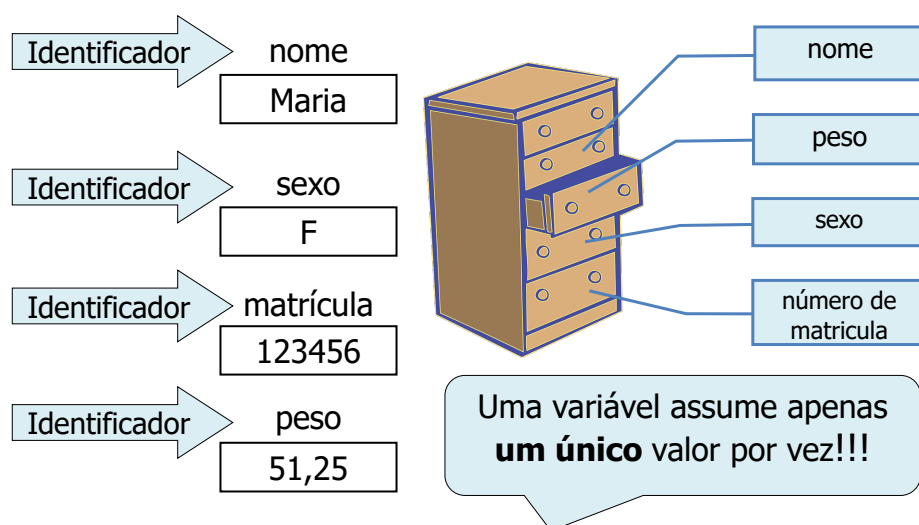
16

## Declaração de Variáveis: Exemplo

- A universidade necessita cadastrar seus alunos, e para isso precisa de um programa que armazene os seguintes dados:
  - nome, sexo, número da matrícula, peso.
- Quais são os passos iniciais ?
  - Definir os tipos: string (**nome**), caractere (**sexo**), inteiro (**número da matrícula**) e real (**peso**).

17

## Variáveis



20



## Identificador: Regras

- Não podem ser iguais a outros identificadores;
- Devem começar por um caractere alfabético;
  - Ex.: x, y, alpha, **5X (identificador inválido)**
- Pode ser seguido de mais caracteres alfabéticos, numéricos ou “\_” (**não vamos usar acentos!**);
  - Ex.: abc, x1, y3, nota\_media
- Não devem ser usados caracteres especiais (#,@,%,?);
  - Ex.: **a\*, A&C, c@sa (identificadores inválidos)**

21

## Identificador: Regras

- Não poder haver espaços em branco entre os caracteres;
  - Ex.: **x 1, I F S C, (identificadores inválidos)**
- Os nomes dos identificadores não podem ser os mesmos das palavras reservadas da linguagem de programação;
  - Ex.: **cos, for, if (identificadores inválidos)**
- **Case sensitivity** ('a' ≠ 'A');

22

## Atribuição de Valores em Python

- Comando de atribuição é o comando que indica que a variável vai receber um valor em seu conteúdo num determinado momento.
- Em Python, o operador de igualdade (=) desempenha esse papel.
- Sempre na forma: **variável = valor** ou **expressão**
  - a expressão do lado direito é processada
  - o valor gerado é atribuído à variável

23

## Atribuição de Valores em Python

- Variável do lado esquerdo e valor do lado direito

```
x = 1
```

- Atribuindo valor a várias variáveis simultaneamente:

```
a = b = c = 3
```

```
# a, b e c terão o valor 3
```

- Podemos também atribuir simultaneamente valores (de tipos) diferentes para variáveis distintas

```
a, b = 1, 7.4
```

```
# a terá o valor 1 e b o valor 7.4
```

24

## Atribuição de Valores em Python

- Troca de valores entre duas variáveis:

```
x, y = 1.3, 5.9
aux = x
x = y
y = aux
```

- Troca sem usar variável auxiliar:

```
x, y = 1.3, 5.9
x, y = y, x
```

25

## Comentários em Python

- Comentários são trechos do programa ignorados pelo interpretador durante a sua execução.
- Em linha simples, começam com o símbolo #  
– tudo na linha após # é ignorado pelo interpretador!
- Use comentários para documentar seus códigos, isso ajuda e facilita o seu entendimento por outras pessoas.
- Comentário em bloco (multilinhas):  
– delimitado no início e fim por """ (3 aspas)

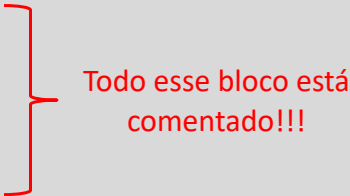
26

## Exemplo

```
# Programa que calcula area de um circulo
pi = 3.1416
raio = 2.1
area = pi * raio * raio

"""
base = 2.5
altura = 7.2
area = base*altura
"""

print(area)
```



27

## Tipos e Variáveis

- Em Python, toda variável tem um tipo
  - não é preciso dizer de que tipo é cada variável!
- Os tipos podem ser divididos em 3 grupos:
  - Numéricos: inteiro, float (real), ...
  - Textuais: caractere e string
  - Lógico (booleano)
- Python possui tipagem dinâmica e forte
- Toda variável é uma referência
  - variáveis armazenam endereços de memória e não valores!

28

## Variáveis Lógicas e Numéricas

```
# Logicas (boolean)
x = True
y = False

# Numericas
a = -8
b = 40
c = 51274145
d = 98.457
e = 0.4e-3
f = 0.791e12
```

29

## Tipagem Dinâmica

a = -8	→	inteiro
b = 40	→	inteiro
c = 51274145	→	inteiro
d = 98.457	→	float
e = 0.4e-3	→	float
f = 0.791e12	→	float

O tipo é determinado **automaticamente** pelo Python no momento de criação da variável

30

## Tipagem Forte

- Uma vez que uma variável tenha um valor de um tipo, ela **não pode** ser usado como se fosse de outro tipo
- Exemplo:

```
>>> x = '1'
>>> y = 2 + x
```

a variável **x** não pode ser somada a um inteiro, pois é um **caractere**

```
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: unsupported operand type(s)
for +: 'int' and 'str'
```

31

## Strings e Caracteres

- Uma string pode ser criada em Python usando aspas simples ou duplas.

```
x = "USP" # ou
y = 'USP'
```

- Exemplos:

```
Nome = "Afonso"
Sobrenome = 'Paiva'
letra = 'Z'
texto = 'Hello World!'
```

32

## Strings e Caracteres

- O operador **+** pode ser usado para concatenar strings ou caracteres.

```
>>> Nome = "Afonso" + " " + "Paiva"
>>> print(Nome)
# o valor de Nome é 'Afonso Paiva'
```

- O operador **\*** pode ser usado para repetir strings ou caracteres.

```
>>> A = 5 * "a"
>>> print(A)
# o valor da variável A é 'aaaaa'
```

33

## Strings e Caracteres

- Python utiliza a tabela de caracteres *default* do **Sistema Operacional**.
  - Exemplos: **ASCII**, **UTF-8**
- Caracteres não imprimíveis podem ser expressos usando o símbolo de *barra invertida* (**\**)
  - **\n** é o mesmo que **nova linha**
  - **\r** é o mesmo que **return**
  - **\t** é o mesmo que **tab**
  - **\b** é o mesmo que **backspace**
  - **\\** é o mesmo que **\**

34

## Strings e Caracteres

- Exemplos usando o **IDLE**:

```
>>> print("Bom dia IFSC - USP")
Bom dia IFSC - USP
>>> print("\tBom dia IFSC - USP")
    Bom dia IFSC - USP
>>> print("Bom dia \nIFSC - USP")
Bom dia
IFSC - USP
>>> print("Bom dia IFSC -\bUSP")
Bom dia IFSC USP
>>> print("Bom dia \rIFSC - USP")
IFSC - USP
>>> print("Bom dia IFSC\\USP")
Bom dia IFSC\USP
```

35

## Strings e Caracteres

- Exemplos:

```
print("Esse texto \
pode ser muito \
grande mesmo")
# uma string escrita em várias linhas com \
```

```
print('''
Um elefante.
Dois elefantes.
Três elefantes.
''')
# quebras de linha usando 3 aspas
```

36



## Strings e Caracteres

- Podemos imprimir uma string com letras maiúsculas e/ou minúsculas usando os seguintes métodos: **lower**, **upper** e **title**.

```
frase = "universidade de são paulo"

frase = frase.upper()
# frase possui 'UNIVERSIDADE DE SÃO PAULO'

frase = frase.lower()
# frase possui 'universidade de são paulo'

frase = frase.title()
# frase possui 'Universidade De São Paulo'
```

37

## Strings e Caracteres

- Remoção de caracteres indesejados ou espaços em brancos no começo ou no fim de uma string pode ser feito com os seguintes métodos: **strip**, **lstrip** e **rstrip**.

```
frase = "    exemplo com string    "
nova_frase = frase.strip()
# remove os espaços no começo e no fim
```

```
frase = "00000exemplo com string00000"
nova_frase = frase.strip('0')
# remove os caracteres 0 no começo e no fim
```

38

## Strings e Caracteres

- Exemplos:

```
frase = "00000exemplo com string00000"
frase1 = frase.lstrip('0')
# remove o caractere 0 no começo (esquerda)
frase2 = frase.rstrip('0')
# remove o caractere 0 no fim (direita)
```

39

## Strings e Caracteres

- A quebra de uma string pode ser feita usando o método **split**.

```
frase = "boa noite"
p1,p2 = frase.split()
# quebra a string nos espaços em branco
# p1 = 'boa' e p2 = 'noite'
```

```
jogo = "vascoXflamengo"
time1,time2 = jogo.split('X')
# quebra a string no caractere X
```

40

## Entrada e Saída de Dados

- A entrada de dados pode ser feita com o comando **input(texto)**
  - *texto* é uma string opcional que será exibida na tela para indicar a entrada de um valor que se espera (i.e., *prompt*).
  - Recebe uma **string como valor de entrada!**

```
x = input()  
cpf= input('Digite seu CPF: ')
```

41

## Entrada e Saída de Dados

- Enquanto que para a saída de dados, usamos o comando **print(expr1,expr2,...)**
  - Os valores das expressões (*expr1,expr2,...*) são escritos na sequência, um após o outro.

```
print(x)  
print("O número do CPF é:", cpf)
```

42

## Entrada e Saída de Dados

- Exemplo:

```
nome = input('Entre com seu nome: ')\npeso = 70\nprint('O peso de', nome, 'é', peso, 'kg')\nprint(84)\nA = 2\nB = 6\nprint(A+B)
```

43

## Entrada e Saída de Dados

- Como faço para saber o tipo que o Python atribuiu a uma variável? Basta usar o comando **type**!

```
lado = input('Entre com o lado de um quadrado: ')\nprint(type(lado))
```

- Como faço para entrar com mais de um valor usando apenas um **input**? Use **split**!

```
a,b = input('Digite 2 números').split()\n# nesse caso os números são separados com espaço
```

44

## Entrada e Saída de Dados

- Exemplo:

```
a,b = input('Digite 2 números inteiros:').split()
a,b = int(a),int(b)
print(a+b)
```

É necessário a conversão de tipos!

45

## Formatação de Saída

- É possível formatar um número pelo comando **print** para que ele seja impresso com um determinado formato
- Exemplo: podemos fazer que um float seja impresso com apenas 2 casas decimais.
  - **print("%.2f" % variável)**

**f** é usado para números do tipo **float**  
**d** é usado para números do tipo **inteiro**  
**s** é usado para o tipo **string**

46

## Formatação de Saída

- Exemplo:
  - Formatação de números

```
x = 9.4657896468
y = 231.12
print("x=%.3f" % x)
print("%.4f" % x)
print("y=%d" % y)
print("%.2f" % x, "%d" % y)
```

```
x = 212.4657896468
x = "%.3f" % x
# cuidado, x virou uma string!
```

O valor de **y** é convertido para **inteiro**.

47

## Formatação de Saída

- Usando uma lista de variáveis :

```
print("%s %d %f" % (x, y, z))
```

- Exemplo:

```
nome = "Pedro"
idade = 21
print("%s tem %d anos." % (nome, idade))
```

48

## Conversão entre Tipos

- A mudança tipos numéricos e strings pode ser feita usando os comandos: **int**, **float**, **str** e **eval**.

```
lado = input('Entre com o lado de um quadrado: ')
print(type(lado))
lado = float(lado)
print(type(lado))
area = lado * lado
print("A área do quadrado é: ", area)
area = str(area)
print(type(area))
Area = "lado * lado"
print(int(eval(Area)))
```

49

## Operadores Aritméticos

Símbolo	Operação	Prioridade
( )	agrupamento	1
**	potenciação	2
-	sinal negativo (unário)	3
+	valor do operando (unário)	3
*	multiplicação	4
/	divisão	4
//	divisão inteira	4
%	resto da divisão inteira	4
+	Adição	5
-	Subtração	5

Em empate de prioridade a **associatividade** é da esquerda para direita

50

## Operadores Aritméticos

- Usando o Python como calculadora via **IDLE**

```
>>> 2*3
```

```
6
```

```
>>> 3**4
```

```
81
```

```
>>> 12//8
```

```
1
```

```
>>> (5 - 1)**(0.5)
```

```
2
```

```
>>> 5/2
```

```
2.5
```

```
>>> 12%8
```

```
4
```

```
>>> 3*(2 + 1)
```

```
9
```

```
>>> 2 + 6/2
```

```
5
```

51

## Operadores Aritméticos

- Usando o Python como calculadora via **IDLE**

```
>>> a = 1
```

```
>>> a = 2*a
```

```
>>> a
```

```
2
```

```
>>> a,b = 3*a,a**2
```

```
>>> a,b
```

```
(6, 4)
```

```
>>> a,b = b,a
```

```
>>> a,b
```

```
(4, 6)
```

52



## Números Complexos

- Representados com dois floats: um para a **parte real** e outro para a **parte imaginária**.
- Escritos como uma soma, sendo que a parte imaginária tem o sufixo **j** ou **J**

53

## Números Complexos

- Exemplo:

```
>>> 1+2j
(1+2j)
>>> 1+2j*3
(1+6j)
>>> (1+2j)*3
(3+6j)
>>> (1+2j)*3j
(-6+3j)
```

54

## Funções Embutidas

- Além dos operadores, é possível usar funções para manipular e computar valores.
- As funções podem ser definidas:
  - pelo programador (veremos mais adiante)
  - em **módulos** da biblioteca padrão
  - por *default*: são as funções *embutidas* (*built-in*)

55

## Funções Embutidas

Função	Descrição
<code>min(x, y)</code>	retorna o mínimo entre x e y
<code>max(x, y)</code>	retorna o máximo entre x e y
<code>round(x)</code>	aproxima para o inteiro mais próximo
<code>abs(x)</code>	valor absoluto de x
<code>chr(x)</code>	retorna um caractere cujo código ASCII é x
<code>ord(x)</code>	retorna o código ASCII do caractere x

A função **abs** preserva o tipo da variável x.

56

## Funções Embutidas

- Exemplos:

```
>>> max(1.3, 4.1)
4.1
>>> abs(-2.3)
2.3
>>> round(2.3)
2
>>> chr(116)
't'
>>> ord('h')
104
```

Decimal	Hex	Oct	Character	Decimal	Hex	Oct	Character	Decimal	Hex	Oct	Character
32	20	040	space	64	40	100	@	96	60	140	^
33	21	041	!	65	41	101	A	97	61	141	a
34	22	042	"	66	42	102	B	98	62	142	b
35	23	043	#	67	43	103	C	99	63	143	c
36	24	044	\$	68	44	104	D	100	64	144	d
37	25	045	%	69	45	105	E	101	65	145	e
38	26	046	&	70	46	106	F	102	66	146	f
39	27	047	'	71	47	107	G	103	67	147	g
40	28	050	(	72	48	110	H	104	68	150	h
41	29	051	)	73	49	111	I	105	69	151	i
42	2A	052	*	74	4A	112	J	106	6A	152	j
43	2B	053	+	75	4B	113	K	107	6B	153	k
44	2C	054	,	76	4C	114	L	108	6C	154	l
45	2D	055	-	77	4D	115	M	109	6D	155	m
46	2E	056	.	78	4E	116	N	110	6E	156	n
47	2F	057	/	79	4F	117	O	111	6F	157	o
48	30	060	0	80	50	120	P	112	70	160	p
49	31	061	1	81	51	121	Q	113	71	161	q
50	32	062	2	82	52	122	R	114	72	162	r
51	33	063	3	83	53	123	S	115	73	163	s
52	34	064	4	84	54	124	T	116	74	164	t
53	35	065	5	85	55	125	U	117	75	165	u
54	36	066	6	86	56	126	V	118	76	166	v
55	37	067	7	87	57	127	W	119	77	167	w
56	38	070	8	88	58	130	X	120	78	170	x
57	39	071	9	89	59	131	Y	121	79	171	y
58	3A	072	:	90	5A	132	Z	122	7A	172	z
59	3B	073	;	91	5B	133	[	123	7B	173	{
60	3C	074	<	92	5C	134	\	124	7C	174	
61	3D	075	=	93	5D	135	]	125	7D	175	}
62	3E	076	>	94	5E	136	^	126	7E	176	~
63	3F	077	?	95	5F	137	_	127	7F	177	DEL

Tabela ASCII

57

## Importando Módulos

- No Python, muitas funções importantes são disponibilizadas em módulos da biblioteca padrão.
- Por exemplo, o módulo **math** possui funções trigonométricas, tais como **sin**, **cos** e a constante matemática **pi**.

58

## Importando Módulos

- Um módulo pode conter não só funções mas também constantes ou classes. Para usar os elementos devemos usar o comando **import** no início do programa.
- Forma Geral:
  - **import** *módulo*
  - **from** *módulo* **import** *nome,...,nome*
  - **from** *módulo* **import** \*

59

## Importando Módulos

- Exemplos:

```
from math import *  
# importa todos os elementos do módulo math
```

```
from math import sin  
# importa apenas a função seno
```

```
import math  
# importa o módulo math como um todo  
# (todos os elementos têm que ser citados  
# precedidos por math.)
```

60

## Funções Científicas: constantes, números e suas representações

Função	Descrição
<code>math.ceil(x)</code>	arredonda para cima
<code>math.fabs(x)</code>	valor absoluto de x (retorna em float)
<code>math.floor(x)</code>	arredonda para baixo
<code>math.fmod(x, y)</code>	resto da divisão de x por em y (float)
<code>math.gcd(x, y)</code>	MDC entre x e y
<code>math.trunc(x)</code>	parte inteira de x
<code>math.pi</code>	constante matemática $\pi = 3.141592...$
<code>math.e</code>	constante matemática $e = 2.718281...$

61

## Funções Científicas: potência e logaritmos

Função	Descrição
<code>math.exp(x)</code>	exponencial de x
<code>math.log(x)</code>	logaritmo natural de x
<code>math.log(x, y)</code>	logaritmo de x na base y
<code>math.log10(x)</code>	logaritmo de x na base 10
<code>math.pow(x, y)</code>	$x^{**}y$
<code>math.sqrt(x)</code>	raiz quadrada de x

62

## Funções Científicas: funções trigonométricas e ângulos

Função	Descrição
<code>math.sin(x)</code>	seno de x
<code>math.asin(x)</code>	arco seno de x
<code>math.cos(x)</code>	cosseno de x
<code>math.acos(x)</code>	arco cosseno de x
<code>math.tan(x)</code>	tangente de x
<code>math.atan(x)</code>	arco tangente de x

Funções trigonométricas trabalham em radianos.

63

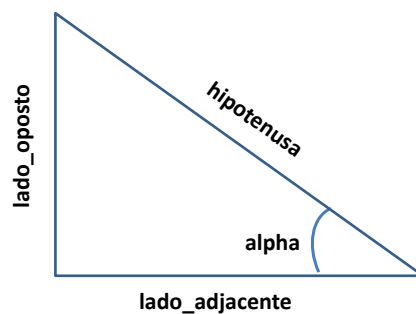
## Funções Científicas: funções trigonométricas e ângulos

Função	Descrição
<code>math.hypot(x, y)</code>	$\sqrt{x^2 + y^2}$
<code>math.degrees(x)</code>	converte radianos para graus
<code>math.radians(x)</code>	converte graus para radianos

64

## Exemplo em Python: lados de um triângulo

Calcular os dois catetos de um triângulo retângulo, dados um ângulo (alpha) em graus e a hipotenusa.



65

## Exemplo em Python: lados de um triângulo

```
import math

hipotenusa = float(input("hipotenusa = "))
alpha = float(input("alpha = "))

ang_rad = math.radians(alpha)
lado_oposto = hipotenusa * math.sin(ang_rad)
lado_adjacente = hipotenusa * math.cos(ang_rad)

print("lado oposto = %.2f" % lado_oposto)
print("lado adjacente = %.2f" % lado_adjacente)
```

66

## Números Aleatórios

- Para gerar um número aleatório no intervalo  $[0,1)$ , precisamos importar o módulo **random**.

```
import random
z = random.random()
# z possui um valor (float) entre 0 e 1
```

- Podemos generalizar a geração de números aleatórios para outros intervalos:

```
z = inicio + (fim - inicio)*random.random()
# z possui um valor entre inicio e fim
```

67

## Números Aleatórios

- Exemplos:

```
print(random.random())
# número entre 0 e 1
```

```
print(random.random() * 10)
# número entre 0 e 10
```

```
print(8 + random.random())
# número entre 8 e 9
```

```
print(30 + random.random() * 60)
# número entre 30 e 90
```

```
x = random.randint(7,14)
# x possui um número inteiro entre 7 e 14
```

68



## Operadores Relacionais

- Utilizados para comparar dois valores do mesmo tipo: sempre retornam valores lógicos **True** ou **False**

Operador	Operação	Prioridade
==	Igual a	6
>	Maior que	6
<	Menor que	6
>=	Maior ou igual a	6
<=	Menor ou igual a	6
!=	Diferente a (≠)	6

prioridade **inferior** aos operadores aritméticos e **associatividade** é da esquerda para direita

69

## Operadores Relacionais

- Conectivos lógicos também sempre retornam valores lógicos **True** ou **False**!

Operador	Operação	Prioridade
not	Não	7
and	E	8
or	OU	9

prioridade **inferior** aos operadores relacionais e muita atenção a **tabela verdade** associada a cada conectivo!

70

## Tabela Verdade para “E” (and)

- **Exemplo:** Se chover e relampejar, eu fico em casa.

- Quando eu fico em casa?

- ❖ p: Está chovendo
- ❖ q: Está relampejando
- ❖  $p \text{ e } q \leftrightarrow p \wedge q$

Conectivo E Conjunção		
p	q	$p \wedge q$
V	V	V
V	F	F
F	V	F
F	F	F

71

## Tabela Verdade para “OU” (or)

- **Exemplo:** Se acabar café ou acabar o açúcar, irei ao mercado.

- Quando irei ao mercado?

- ❖ p: Acabou o café
- ❖ q: Acabou o açúcar
- ❖  $p \text{ ou } q \leftrightarrow p \vee q$

Conectivo OU Disjunção		
p	q	$p \vee q$
V	V	V
V	F	V
F	V	V
F	F	F

72

## Operadores Relacionais

- Exemplos:

```
>>> 1==1
```

```
True
```

```
>>> 1==2
```

```
False
```

```
>>> 1==1 or 1==2
```

```
True
```

```
>>> 1==1 and 1==2
```

```
False
```

```
>>> 1<2 and 2<3
```

```
True
```

```
>>> not 1<2
```

```
False
```

```
>>> not 1<2 and 2<3
```

```
False
```

```
>>> not (1<2 or 2<3)
```

```
False
```

73

## Operadores Relacionais

- Qualquer valor **não nulo** é visto como **True**, enquanto que **0** é visto como **False**.
- O operador **or** retorna o primeiro operando só se ele for **True**, caso contrário retorna o segundo.
- O operador **and** retorna o primeiro operando só se ele for **False**, caso contrário retorna o segundo.

74

## Operadores Relacionais

- Exemplos:

```
>>> 0 or 100
100
>>> False or 100
100
>>> "abc" or 1
'abc'
>>> 1 and 2
2
```

```
>>> 0 and 3
0
>>> False and 3
False
>>> 1 and 2 or 3
2
>>> not 1 or 2 and 3
3
```

75

## Operadores de Atribuição

Símbolo	Operação	Equivalente
<code>+=</code>	<code>a += b</code>	<code>a = a+b</code>
<code>-=</code>	<code>a -= b</code>	<code>a = a-b</code>
<code>*=</code>	<code>a *= b</code>	<code>a = a*b</code>
<code>/=</code>	<code>a /= b</code>	<code>a = a/b</code>
<code>**=</code>	<code>a **= b</code>	<code>a = a**b</code>
<code>//=</code>	<code>a //= b</code>	<code>a = a//b</code>
<code>%=</code>	<code>a %= b</code>	<code>a = a%b</code>

76

## Operadores de Atribuição

- Exemplos:

```
>>> a = 3
>>> b = 2
>>> b *= a
>>> b
6
>>> a -= 1
>>> a
2
```