



@PETTELEUFF

Mini Curso Python

Aula 01



AULA 01

Tópicos

1

Informações

2

Introdução à linguagem

3

Python

4

Operadores

5

Estruturas de Decisão

6

Estruturas de Repetição



1

Informações

Sobre o mini curso

- Ministrado por:
 - Integrantes do GRUPO PET-Tele
- OBEJTIVO:
 - Aprender o básico da linguagem Python;
- DURAÇÃO:
 - 4 horas.

Vantagens

- Sintaxe básica, semelhante à do inglês;
- Aumento de produtividade
 - menos linha de código comparado com outras linguagens
- Automação com scripts Python
 - Renomeação de um grande número de arquivos de uma vez só;
 - Converter um arquivo em outro tipo;
 - Remoção de palavras duplicadas em arquivo de texto;
 - Execução de operações básicas matemáticas;
 - etc.

2

Introdução à linguagem

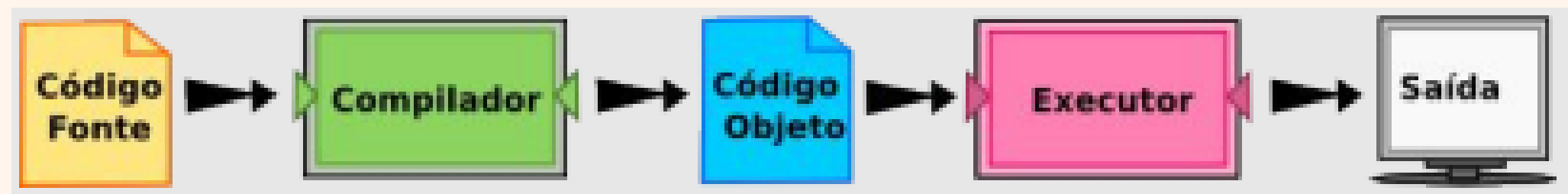
- Python é uma linguagem de programação de alto nível.
- O computador só consegue executar programas escritos em linguagens de baixo nível (“línguas de máquina” ou “linguagens assembly”).
- Precisa ser processado antes de rodar
- Programas de alto nível para baixo nível:
 - **interpretadores**: faz o que o programa diz



2

Introdução à linguagem

- **compiladores:** lê o programa e o traduz antes que comece a rodar;



2

Introdução à linguagem

- O programa traduzido é chamado de código objeto ou executável.
- O Python usa ambos os processos, mas ela é em geral considerada uma linguagem interpretada.
- Existe duas maneiras de usar o interpretador:
 - a. linha de comando ("shell mode")
 - você digita comandos e o interpretador mostra os resultados
 - b. script ("program mode")
 - você escreve um programa inteiro em um arquivo e usa o interpretador para executar o conteúdo do arquivo como todo.

3

Python

IDE – Integrated Development Environment (ambiente de desenvolvimento integrado)

PyCharm



PyCharm Community Edition 2022.2.1

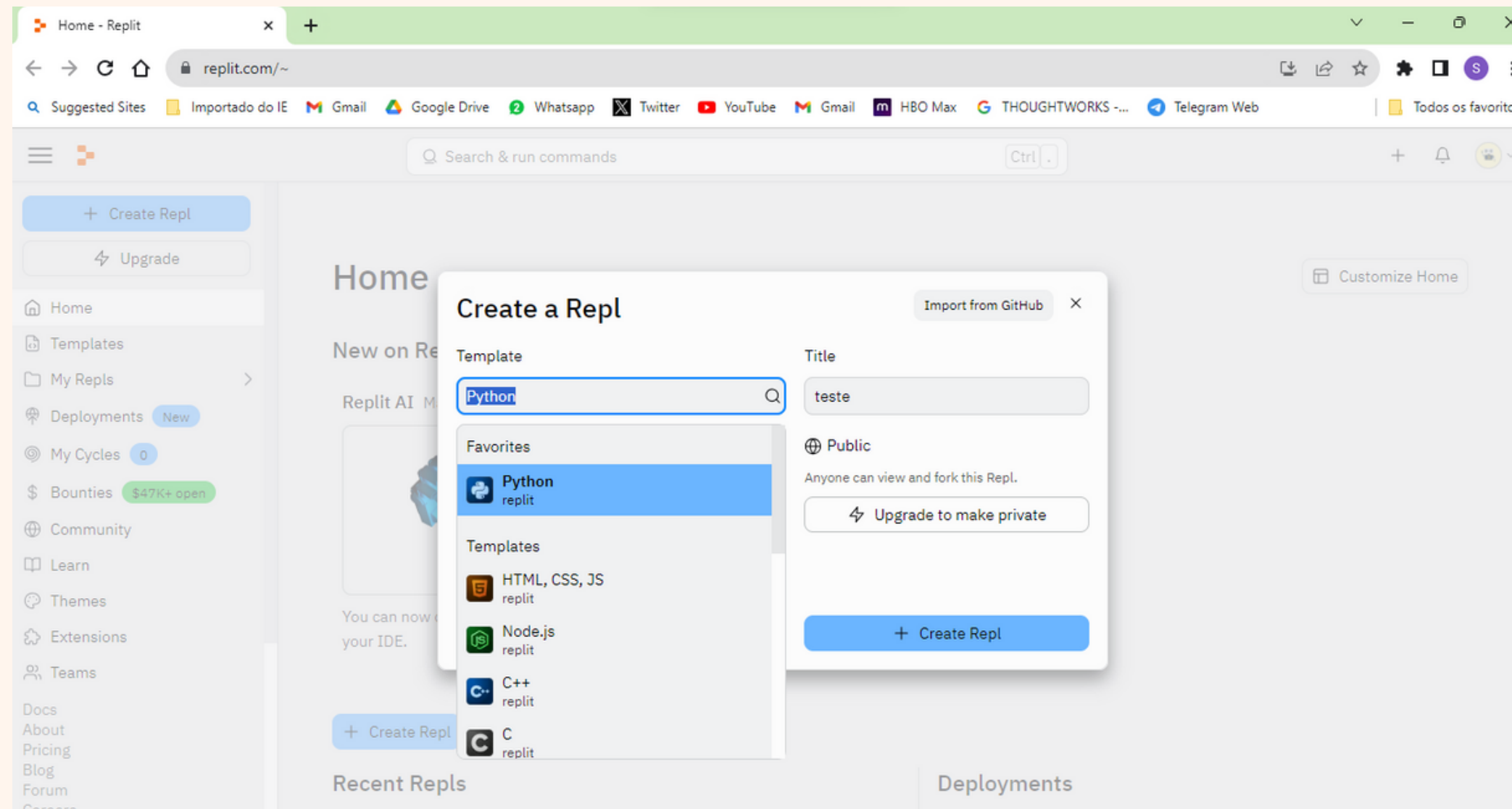
Aplicativo

3

Python

IDE – Integrated Development Environment (ambiente de desenvolvimento integrado)

Replit



3

Python – INFORMAÇÕES BÁSICAS

- Sequência de comando;
 - Os comandos são lidos de cima para baixo;
- Em códigos alternativos ou que se repetem podemos usar blocos de comandos.
 - Esses blocos devem estar indentados! (Usar o tab)
- Comentários (#)
 - Tudo que for escrito após o símbolo # na linha do código será ignorado pelo interpretador;
 - Ferramenta importante para a documentação da escrita do código



ATRIBUIÇÃO DE VALORES

- O operador da igualdade (=) é usado para a atribuição de valores.
 - Variavel = valor ou expressão;
 - expressão do lado direito e processada;
 - valor gerado é atribuído à variável

3

Python

```
d = 3  
c = 2  
b = 4  
d = c + b  
a = d + 1  
a = a + 1  
print(a)
```

Qual valor de a?

3

Python

```
d = 3
c = 2
b = 4
d = c + b
a = d + 1
a = a + 1
print(a)
```

```
d=2+4
d=6
a=6+1
a=7+1
a=8
saida do prog é 8
```

3

Python

```
d = 3.0  
c = 2.5  
b = 4  
d = b + 90  
e = c * d  
a = a + 1  
print(a)  
print(e)
```

3

Python

```
d = 3.0
```

```
c = 2.5
```

```
b = 4
```

```
d = b + 90
```

```
e = c * d
```

```
a = a + 1
```

```
print(a)
```

nenhum valor atribuído no a

```
print(e)
```

o programa não irá rodar!

Python – Tipos de dados

- Toda variável tem um tipo;
 - Tipos numéricos
 - `int`
 - (números inteiros ex: -8, 1, 0, 201558)
 - `float`
 - (ponto flutuante ex: -4.5, 1.0, 1e8.56)
 - Tipos textuais
 - `str`
 - (sequência de caractere/strings ex: “Pet-Tele” , “Setel”)
 - Tipo lógico
 - `bool`
 - (valores booleanos: True e False)

Os tipos são dinamicamente no Python, ou seja, não é necessário determinar o tipo de cada variável na hora de criar-las.

3

Python - Tipos de dados

- Comando type:

main.py > ...

```
1 altura = input('Digite a altura do triangulo: ')\n2 print(type(altura))
```

```
python3 main.py\nDigite a altura do triangulo: 3\n<class 'str'>\n█
```

Mudança de tipos

- Em algumas situações o programador deseja transformar o tipo de uma expressão
 - Para isso, basta envolver a expressão a ser transformada por “tipo(expressão)”
- Exemplo: transformar um real em um inteiro

```
a = 5.1
```

```
x = int(a) # x vale 5
```

- Exemplo: transformar um inteiro em um real

```
b = 5
```

```
y = float(b) # y vale 5.0
```

Tipagem Dinâmica

- Uma variável em Python possui o tipo correspondente ao objeto que ela está associada naquele instante.
- Python não possui tipagem forte como outras linguagens.
 - Isto significa que você pode atribuir objetos de diferentes tipos para uma mesma variável.
 - Como uma variável não possui tipo pré-definido, dizemos que Python tem tipagem fraca.
 - Em outras linguagens cria-se variáveis de tipos específicos e elas só podem armazenar valores daquele tipo para o qual foram criadas.
 - Estas últimas linguagens possuem tipagem forte.

3

Python

Tipagem Dinâmica

`a= -5 -> int`

`b= 2.5 -> float`

`c="hello" -> str`

Os tipos são dinamicamente no Python, ou seja, não é necessário determinar o tipo de cada variável na hora de criar-las.

3

Python

Tipagem Forte

- Se a variável tem um tipo em seu valor, ela não pode ser usada como outro tipo

```
a= 10
```

```
b="10"
```

```
c= a + b
```

```
> python3 main.py
Traceback (most recent call last):
  File "/home/runner/teste/main.py", line 3, in <module>
    c=a+b
TypeError: unsupported operand type(s) for +: 'int' and 'str'
exit status 1
```

3

Python

Tipagem Forte

- Se a variável tem um tipo em seu valor, ela não pode ser usada como outro

Agora vejamos este exemplo em JavaScript:



```
const num = "10";  
const operacao = num + 10; //1010
```

```
TypeError: unsupported operand type(s) for +: 'int' and 'str'  
exit status 1
```

Regras para nomear variáveis

- Nomes de variáveis devem começar com uma letra (maiúscula ou minúscula) ou um sublinhado (_);
- NUNCA pode começar com um número;
- Não pode utilizar no nome da variável: : { (+ - * / \ n ; . , ? \$
- Não podem ter espaços nem acentos;
- São sensíveis a caixa, ou seja, há diferenças se a variável for “A” e “a”;
- Não podem ser palavras reservadas na linguagem;
 - definem a sintaxe da linguagem e sua estrutura.

3

Python

Regras para nomear variáveis

and	as	assert	break	class	continue
def	del	elif	else	except	exec
finally	for	from	global	if	import
in	is	lambda	nonlocal	not	or
ass	raise	return	try	while	with
yield	True	False	None		

Entrada de Dados

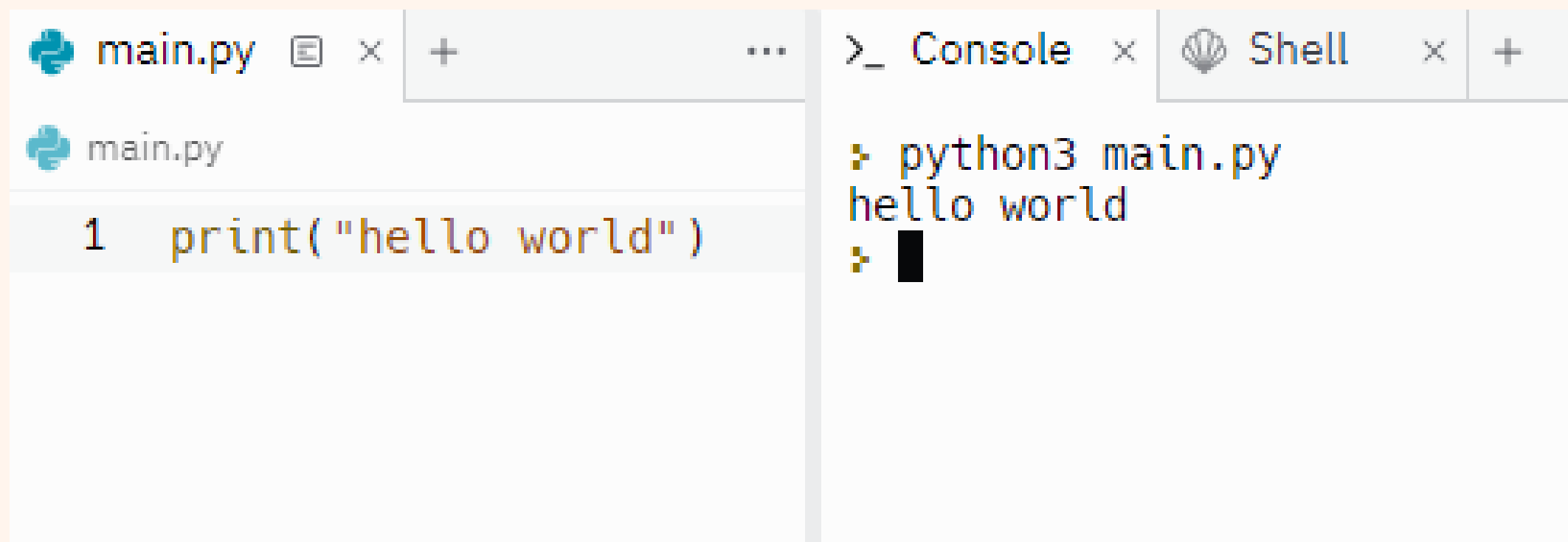
- A função input é responsável por receber dados do usuário.
- O usuário deve escrever algo e pressionar a tecla .
- Normalmente, armazenamos o valor lido em uma variável.
- A função input obtém os dados fornecidos pelo console no formato de string (str).
- Devemos fazer uma conversão dos dados se quisermos trabalhar com números.

3

Python

Saída de dados

- print



The image shows a code editor window with a file named `main.py` containing the following code:

```
1 print("hello world")
```

To the right of the code editor is a terminal window with the following output:

```
> python3 main.py  
hello world  
>
```

3

Python

EXEMPLO

Faça um programa que leia o nome, a idade, a altura, o peso e a nacionalidade do usuário e escreva essas informações na forma de um parágrafo de apresentação

3

Python

EXEMPLO

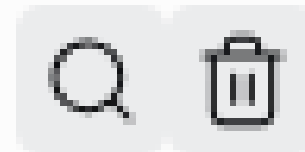
```
1  #Faça um programa que leia o nome, a idade, a altura,  
   o peso e a nacionalidade do usuário e escreva essas  
   informações na forma de um parágrafo de apresentação  
2  nome = input('digite seu nome: ') #entrada de dados,  
   nao eh necessario botar o tipo string nessa entrada  
3  idade = (int(input('digite sua idade: '))) #entrada  
   com o num inteiro  
4  altura = (float(input('digite sua altura: ')))  
   #entrada com num flutuante  
5  peso = (float(input('digite seu peso: ')))  
6  nacionalidade = input('digite sua nacionalidade: ')  
7  print ('Meu nome é', nome , 'tenho',  
   idade, 'anos, ', 'tenho', altura, 'm, ', 'peso', peso,  
   'kg, ', 'sou', nacionalidade) #saida de dados
```

3

Python

EXEMPLO

```
➤ python3 main.py
digite seu nome: Silvana
digite sua idade: 21
digite sua altura: 1.53
digite seu peso: 51.8
digite sua nacionalidade: brasileira
Meu nome é Silvana tenho 21 anos, tenho 1.53 m, peso
51.8 kg, sou brasileira
➤ █
```



3

Python

EXERCÍCIO

Faça um programa que informe a distância em quilômetros de um raio para o observador

- O observador deve informar o tempo (em segundos) transcorrido entre ver o raio e ouvir o trovão
- Assuma que a velocidade do som é 340 m/s

Operadores Aritméticos

Operador	Exemplo	Prioridade
(x)	$(1 + 2) * 3 \rightarrow 9$	1
**	$2 ** 3 \rightarrow 8$	2
+x	+15	3
-x	$-(5+3) \rightarrow -8$	3
*	$5 * 3 \rightarrow 15$	4
/	$5 / 3 \rightarrow 1.66$	4
//	$5 // 3 \rightarrow 1$	4
%	$5 \% 3 \rightarrow 2$	4
+	$5 + 3 \rightarrow 8$	5
-	$5 - 3 \rightarrow 2$	5

Operadores Aritméticos

- Operadores com a mesma prioridade (precedência)
- Analisados da esquerda para a direita
- Divisão de inteiros (`//`)
 - Resultado é somente a parte inteira da divisão
- Divisão (`/`)
 - Resultado fracionário

- Antes de escrever o programa SEMPRE que quiser usar as funções tem que chama-las no início

- `import math`

- Constantes

- `math.pi` -> 3.1415...
 - `math.e` -> 2.7182...

função	Descrição	Exemplo
<code>abs(x)</code>	Valor absoluto	<code>abs(-5.3)</code> → 5.3
<code>round(x, y)</code>	Arredonda x em y dígitos	<code>round(2.677, 2)</code> → 2.68
<code>math.ceil(x)</code>	Arredonda para cima	<code>math.ceil(5.3)</code> → 6
<code>math.floor(expr)</code>	Arredonda para baixo	<code>math.floor(5.3)</code> → 5
<code>max(n1, n2, ...)</code>	Maior dentre vários números	<code>max(3, 4, 5)</code> → 5
<code>min(n1, n2, ...)</code>	Menor dentre vários números	<code>min(3, 4, 5)</code> → 3
<code>math.sqrt(x)</code>	Raiz quadrada	<code>math.sqrt(16)</code> → 4.0
<code>math.log(x, y)</code>	Logaritmo de x na base y	<code>math.log(2, 10)</code> → 0.3010299...
<code>math.log(x)</code>	Logaritmo natural (base e)	<code>math.log(2)</code> → 0.69314718...
<code>math.exp(x)</code>	$e^{**}x$	<code>math.exp(2)</code> → 7.38905609...

Funções matemáticas

Trigonométricas

Função	Descrição	Exemplo
<code>math.sin(x)</code>	Seno	<code>math.sin(0) → 0.0</code>
<code>math.asin(x)</code>	Arco seno	<code>math.asin(1) → 1.5707963267948966</code>
<code>math.cos(x)</code>	Cosseno	<code>math.cos(0) → 1.0</code>
<code>math.acos(x)</code>	Arco cosseno	<code>math.acos(-1) → 3.141592653589793</code>
<code>math.tan(x)</code>	Tangente	<code>math.tan(1) → 1.5574077246549023</code>
<code>math.atan(x)</code>	Arco tangente	<code>math.atan(1) → 0.7853981633974483</code>
<code>math.degrees(x)</code>	Converte radianos para graus	<code>math.degrees(math.pi) → 180.0</code>
<code>math.radians(x)</code>	Converte graus para radianos	<code>math.radians(180) → 3.141592653589793</code>

Números aleatórios

- Sorteio de números
 - Função:
 - `random.random()`
 - Gera números pseudo aleatório no intervalo de [0.1)
- A partir disso, é possível gerar números em outros intervalos
 - `inicio + (fim - inicio) * random.random()`
- `import random`

Geração de números aleatórios inteiros

```
import random  
x = random.randint(3, 9)  
# x conterá um número inteiro sorteado  
# entre 3 e 9, inclusive
```

Operador	Exemplo	Prioridade
$x < y$	$5 < 3 \rightarrow \text{False}$	6
$x \leq y$	$5 \leq 3 \rightarrow \text{False}$	6
$x > y$	$5 > 3 \rightarrow \text{True}$	6
$x \geq y$	$5 \geq 3 \rightarrow \text{True}$	6
$x == y$	$5 == 3 \rightarrow \text{False}$	6
$x != y$	$5 != 3 \rightarrow \text{True}$	6

- Prioridade sempre inferior aos operadores aritméticos
- Sempre têm números ou strings como operandos
- Sempre têm resultado booleano

Operadores Lógicos

Operador	Exemplo	Prioridade
not x	not True → False	7
x and y	True and False → False	8
x or y	True or False → True	9

- Prioridade sempre inferior aos operadores relacionais
- Sempre têm booleanos como operandos
- Sempre têm resultado booleano

tabela da verdade

a	b	not a	a and b	a or b
True	True	False	True	True
True	False	False	False	True
False	True	True	False	True
False	False	True	False	False

Operadores de Atribuição

Operador	Exemplo
<code>var = expr</code>	<code>x = 10 + 5</code>
<code>var += expr</code>	<code>x += 5 → x = x + 5</code>
<code>var -= expr</code>	<code>x -= 5 → x = x - 5</code>
<code>var *= expr</code>	<code>x *= 5 → x = x * 5</code>
<code>var /= expr</code>	<code>x /= 5 → x = x / 5</code>
<code>var //= expr</code>	<code>x //= 5 → x = x // 5</code>
<code>var %= expr</code>	<code>x %= 5 → x = x % 5</code>
<code>var **= expr</code>	<code>x **= 5 → x = x ** 5</code>

5

Estruturas de Decisão

- Os comandos de Python são executados pelo computador, linha por linha e as estruturas de controle permitem ao programador modificar a ordem em que cada comando será executado bem como se ele será ou não executado.

Decisão *if*

- Executa o bloco o bloco de instruções somente se a condição for verdadeira;
- condição é uma expressão booleana;
- bloco é delimitado por indentação.

```
1  op = input('Deseja somar? (S/N)')
2  if (op == 'S'):
3      x = float(input('Digite o primeiro numero:'))
4      y = float(input('Digite o segundo numero:'))
5      resultado = x + y
6      print('O resultado da soma é', resultado)
7  print('Até a próxima!')
```

if... else

- Executa um ou o outro bloco de instrução em função da condição ser verdadeira ou falsa;
- condição é uma expressão booleana;
- bloco é delimitado por indentação.

```
1  numero = int(input('Entre com um número: '))
2  if numero % 2 == 0:
3      print('O número é par.')
4  else:
5      print('O número é impar.')
```

if... else

Programa para somar ou multiplicar dois numeros

```
1 op = input('Deseja somar (S) ou multiplicar (M)? ')
2 ✓ if (op == 'S' or op == 'M'):
3     x = float(input('Digite o primeiro numero:'))
4     y = float(input('Digite o segundo numero:'))
5 ✓     if (op == 'S'):
6         r = x + y
7 ✓     else:
8         r = x * y
9         print('O resultado é', r)
10 ✓ else:
11     print('Opção inválida!')
```

if...elif

- Executa apenas o bloco no qual a condição é verdadeira;
- Pode botar quantos ***elif*** forem necessários.
- É possível adicionar um ***else*** ao final de tudo
 - se nenhuma condição for verdadeira o bloco ***else*** será executado.

if...elif

```
1 mes = int(input('Entre com um mês (1 a 12): '))
2
3 ✓ if (mes==1 or mes==3 or mes==5 or mes==7 or mes==8 or mes==10 or
   mes==12):
4     print('Esse mes tem 31 dias.')
5 ✓ elif mes==4 or mes==6 or mes==9 or mes==11:
6     print('Esse mes tem 30 dias.')
7 ✓ elif mes==2:
8     ano = int(input('Entre com o ano (4 dígitos): '))
9 ✓     if ano % 400 == 0 or ano % 4 == 0 and ano % 100 != 0:
10        print('Esse mes tem 29 dias.')
11 ✓     else:
12        print('Esse mes tem 28 dias.')
13
14 ✓ else:
15     print('Mês Inválido!')
16
```

Generate Alt G

- Permite que o bloco de comando seja executado diversas vezes;
- Dois tipos de repetição:
 - **Condicional** (***while***): executa um bloco de código enquanto uma condição lógica for verdadeira;
 - **Contável** (***for***): executa um bloco de código num número predeterminado de vezes

Repetição condicional

while

- Executa o bloco de instrução enquanto a condição for verdadeira;
 - igual ao ***if***
- A estrutura de repetição é chamada de loop porque continua-se voltando ao início da instrução até que a condição se torne falsa
- Deve haver alguma instrução dentro do bloco de comandos que torne a condição falsa para que a repetição seja encerrada
- Quando a condição se torna falsa, a próxima instrução após o bloco do ***while*** é executada;
- Se a condição do ***while*** for falsa desde o início, o bloco de instruções nunca é executado.

Repetição condicional

while

```
1  numero = int(input('Digite um número: '))
2  while numero > 0:
3      numero = numero - 1
4      print(numero)
5  print('Boom!!!')
6  |
```

Generate Alt G

Repetição condicional

while

```
➤ python3 main.py  
Digite um número: 5  
4  
3  
2  
1  
0  
Boom!!!  
➤ █
```

Repetição contável

for

faixa de valores

- Os valores podem ser especificados como um intervalo com início, fim e incremento, usando o range

```
for i in range (0 , 5 , 1) :  
    print(i)
```

início (opcional)
quando omitido,
início = 0

fim (obrigatório)

incremento (opcional)
quando omitido,
incremento = 1

for

```
for i in range(1, 5):  
    print(i)
```

```
1  
2  
3  
4
```

Repetição contável

for

```
for i in range(5):  
    print(i)
```

0

1

2

3

4

for

```
for i in range(2, 10, 2):  
    print(i)
```

2

4

6

8

for

```
for i in range(10, 0, -2):  
    print(i)
```

10

8

6

4

2

programa de soma de valores aleatórios com *while*

```
1 import random
2
3 x = int(input('Digite um número: '))
4 soma = 0
5 contador = 0
6 while contador < x:
7
8     numero_sorteado = random.randint(1,10)
9     print(numero_sorteado)
10    soma = soma + numero_sorteado
11    contador = contador + 1
12
13 print('A soma é', soma)
```

Edit Alt T

while

```
➤ python3 main.py  
Digite um número: 5  
7  
9  
3  
1  
5  
A soma é 25  
➤ █
```

programa de soma de valores aleatórios com *for*

```
1 import random
2
3 x = int(input('Digite um número: '))
4 soma = 0
5 for contador in range(x):
6
7     numero_sorteado = random.randint(1,10)
8     print(numero_sorteado)
9     soma = soma + numero_sorteado
10
11 print('A soma é', soma)
```

for

```
➤ python3 main.py  
Digite um número: 5  
5  
1  
5  
2  
3  
A soma é 16  
➤ □
```

EXERCÍCIO

#Faça um programa para determinar o número de dígitos de um número inteiro positivo informado