

# Introdução

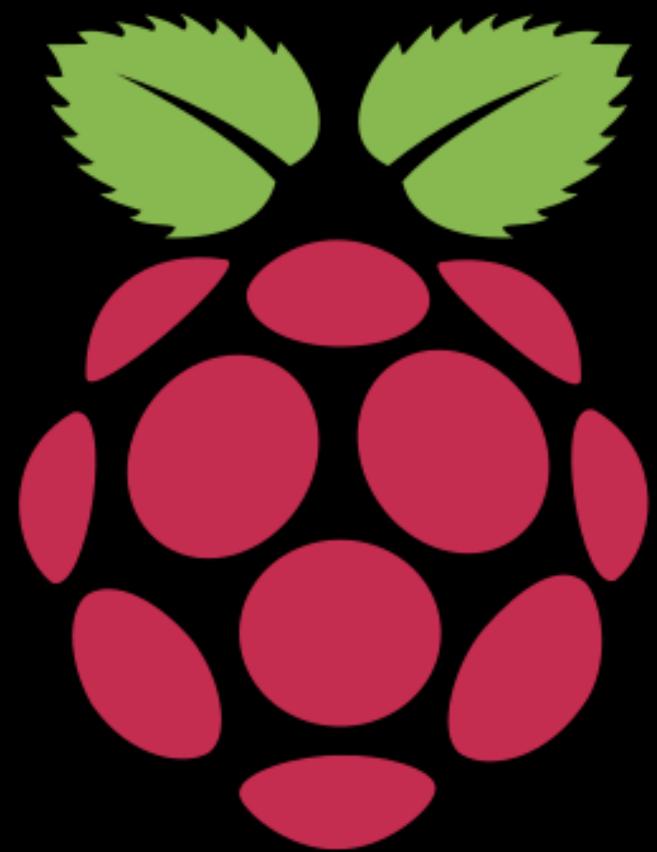
## Revisão de Python – Teoria

Jan K. S. – [janks@puc-rio.br](mailto:janks@puc-rio.br)

ENG 1419 – Programação de Microcontroladores



Prazer, Jan



PUC  
RIO

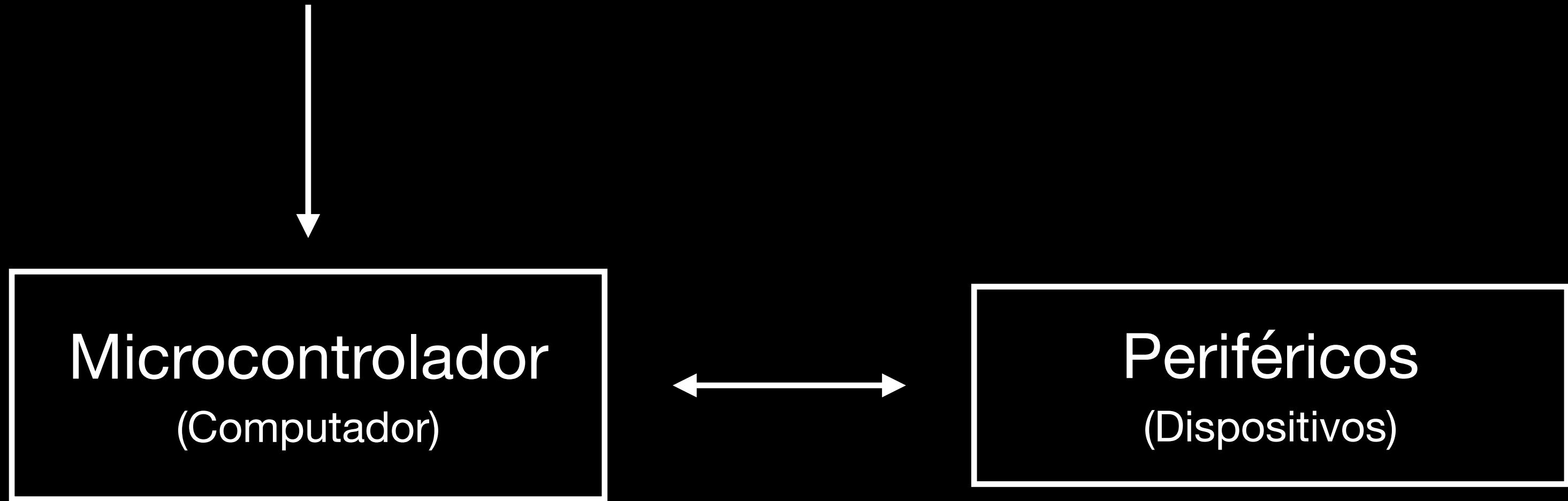
Disciplina de Programação de Microcontroladores na PUC-Rio

O que é um  
microcontrolador?

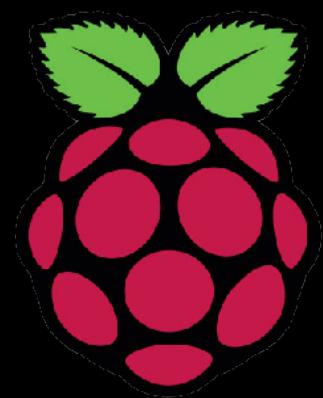
“Microcontrolador é um **pequeno computador** (SoC) em um único circuito integrado o qual contém um núcleo de processador, memória e **periféricos programáveis de entrada e saída**.”

Wikipedia

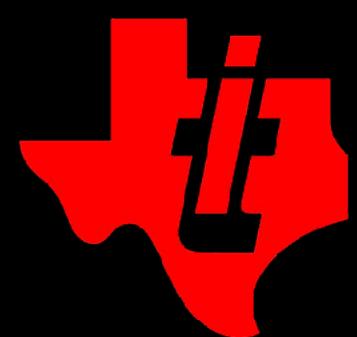
# Programação



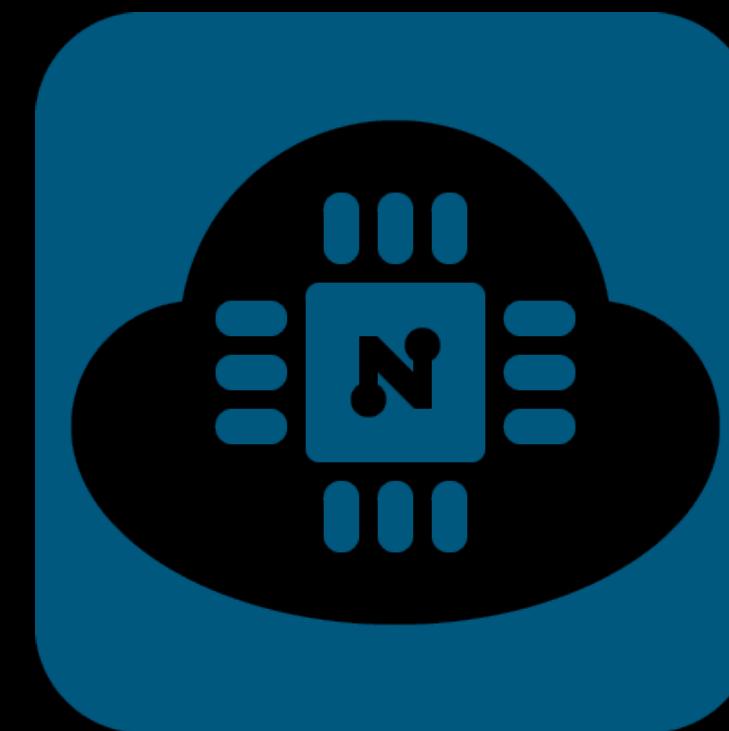
Microcontrolador e Periféricos (Dispositivos) Programaveis



Raspberry Pi



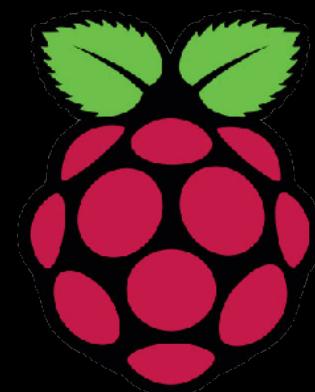
TEXAS  
INSTRUMENTS



Exemplos de Microcontroladores



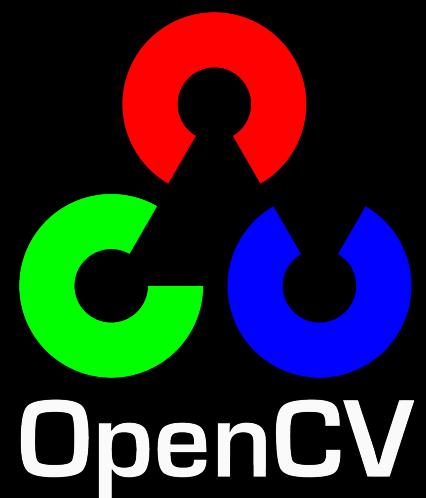
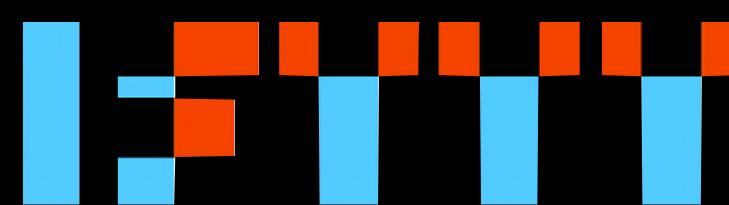
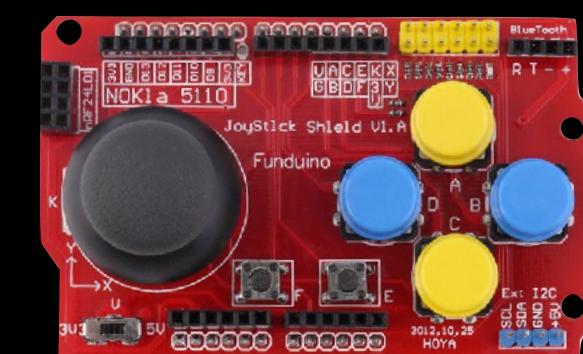
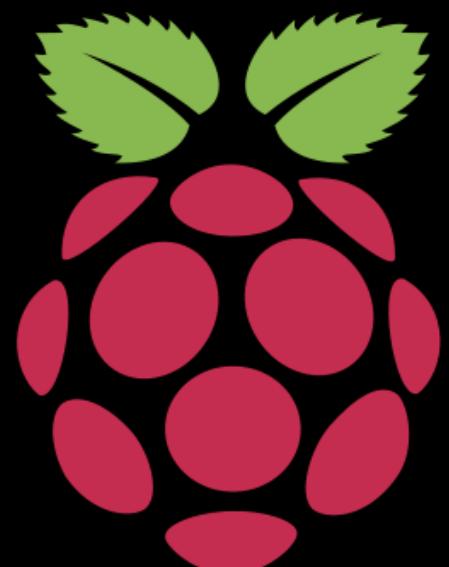
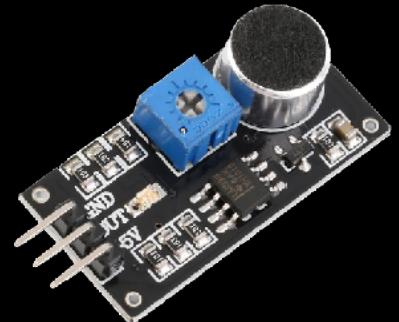
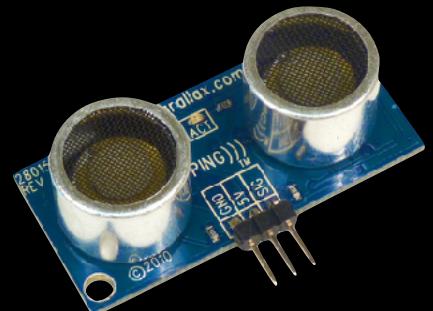
PY



Raspberry Pi



Microcontroladores e Linguagens Usados no Curso



Exemplos de Hardware e Software Vistos no Curso

# Raspberry Pi

Raspberry Pi – Wikipédia, a encyclopédia livre

Não autenticado Discussão Contribuições Criar uma conta Entrar

# Raspberry Pi

Origem: Wikipédia, a encyclopédia livre.

**[[wiki]]** Esta página ou seção precisa ser [wikificada](#) (desde janeiro de 2018). Por favor ajude a [formatar](#) esta página de acordo com as [diretrizes](#) estabelecidas.

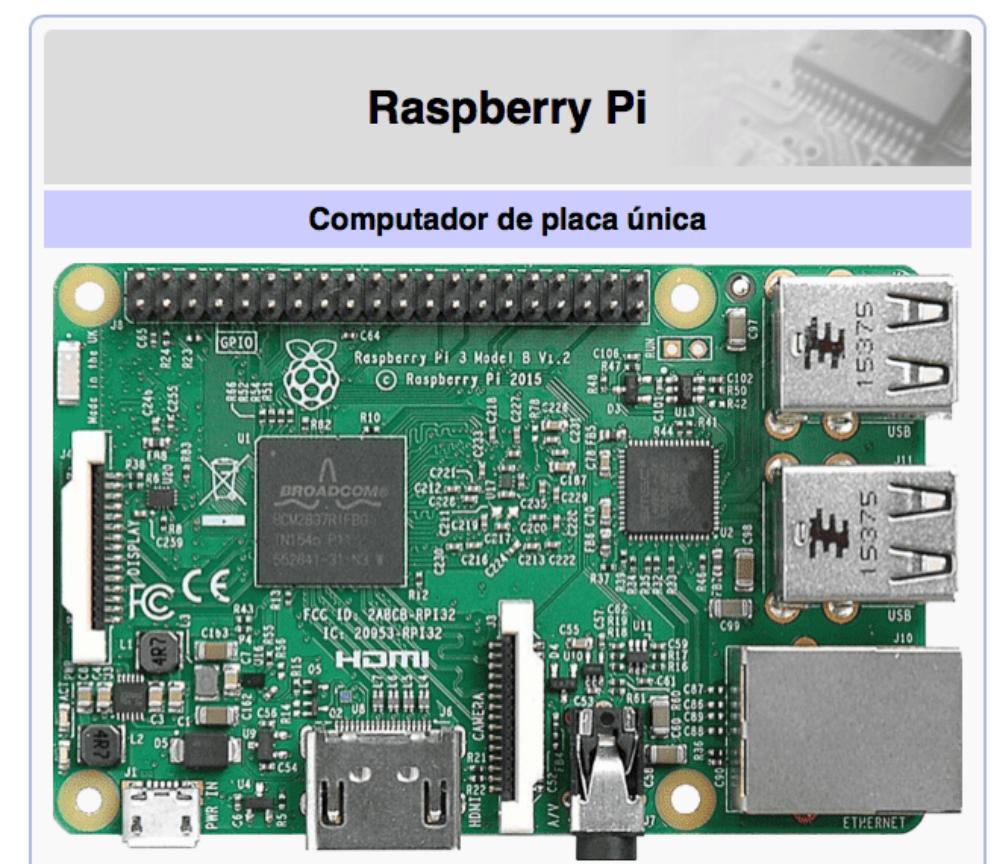
 As referências deste artigo [necessitam de formatação](#) (desde março de 2017). Por favor, utilize [fontes](#) apropriadas contendo referência ao título, autor, data e fonte de publicação do trabalho para que o artigo permaneça [verificável](#) no futuro.

**Raspberry Pi** é um [computador](#) do tamanho de um cartão de crédito, que se conecta a um monitor de computador ou TV, e usa um teclado e um mouse padrão, desenvolvido no [Reino Unido](#) pela [Fundação Raspberry Pi](#). Todo o hardware é integrado numa única placa. O principal objetivo é promover o ensino em [Ciência da Computação](#) básica em escolas.<sup>[1][2][3][4]</sup>

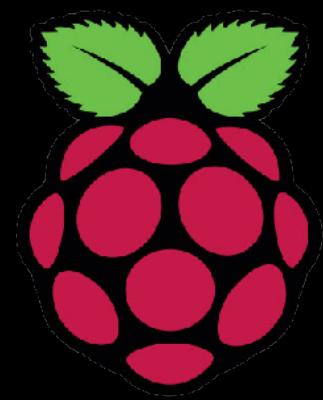
A Fundação Raspberry Pi começou a aceitar pedidos do modelo de US\$35 a partir de 29 de fevereiro de 2012.<sup>[5]</sup>

O Raspberry Pi 1 é baseado em um [system on a chip](#) (SoC) [Broadcom BCM2835](#), que inclui um processador [ARM1176JZF-S](#) de 700 MHz, [GPU](#) VideoCore IV, e 512 MB de [memória RAM](#) em sua última revisão. O projeto não inclui uma memória não-volátil - como um [disco rígido](#) - mas possui uma entrada de [cartão SD](#) para armazenamento de dados.<sup>[6]</sup>

O Raspberry Pi 3 model B contém um processador 1.2GHz 64-bit quad-core ARMv8 CPU, 1 GB de RAM, Bluetooth 4.1.

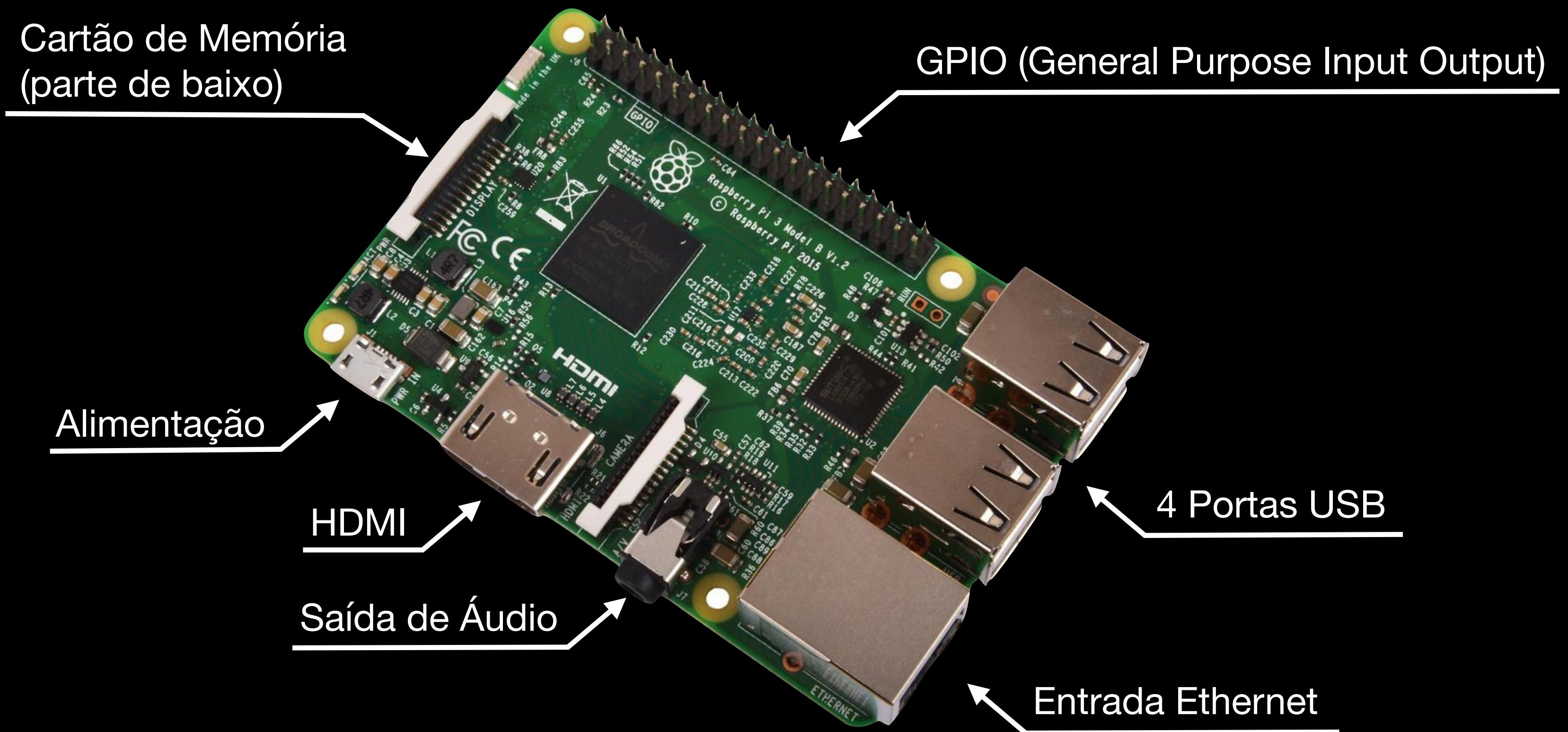


Dados sobre o Raspberry Pi

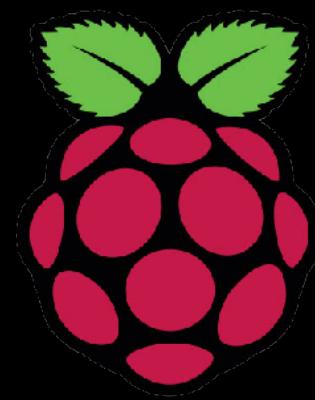


# Raspberry Pi

(Modelo 3 B)



Portas de Conexão do Hardware



# Raspberry Pi

(Modelo 3 B)



Processador ARMv8 1.2GHz de 4 núcleos

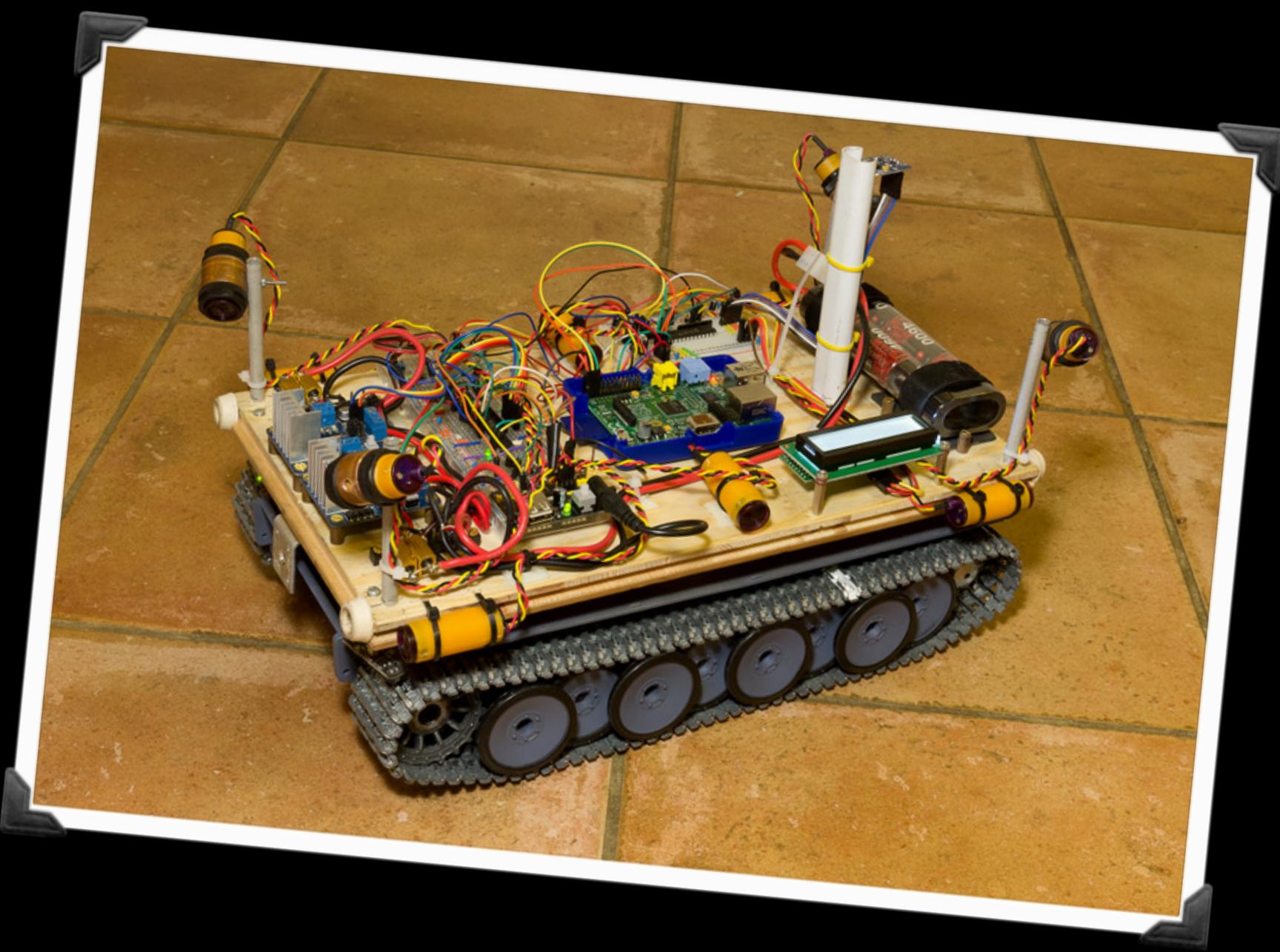
GPU Broadcom VideoCore IV

1GB de RAM (LPDDR2, 900 MHz)

WiFi 2.4GHz 802.11n

Bluetooth 4.1

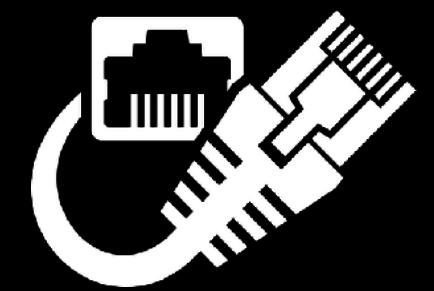
Especificações Técnicas Internas



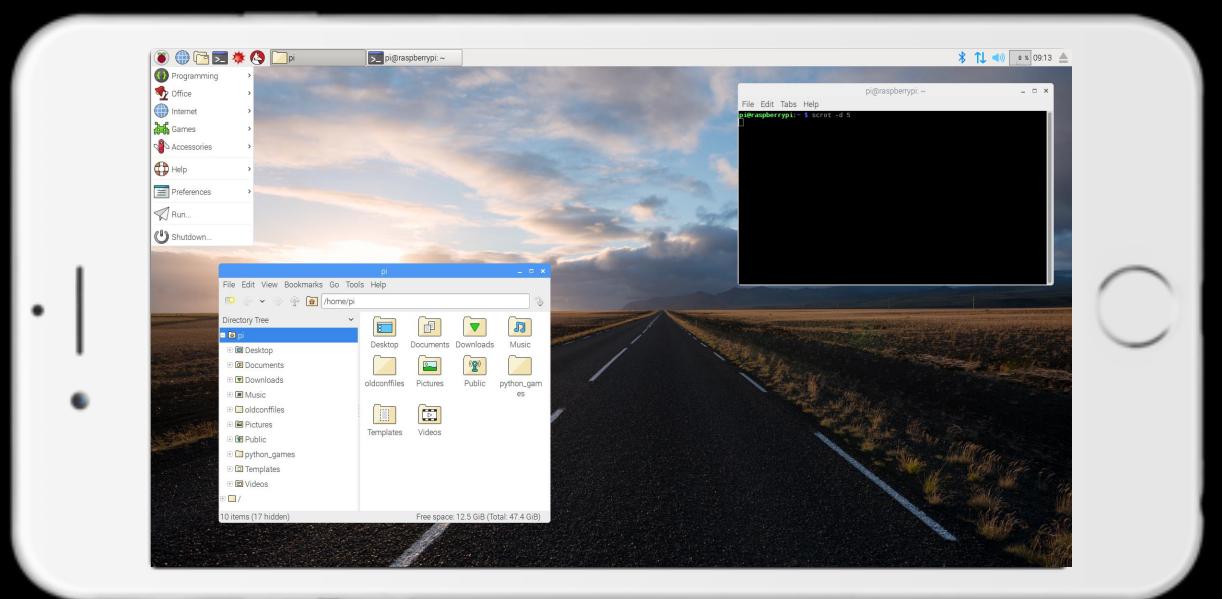
Exemplos de Aplicações com o Raspberry Pi



Opção 1: Acesso Local (Monitor + Mouse + Teclado)



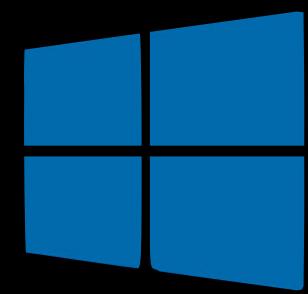
Opção 2: Acesso Remoto



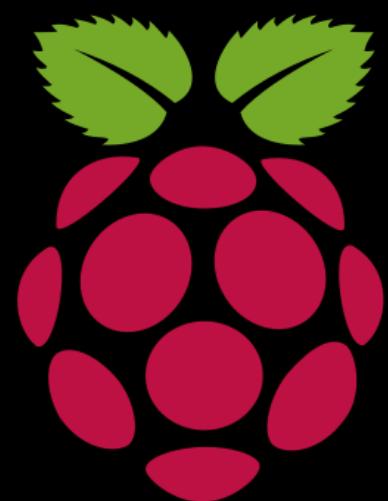
Acesso Remoto via Real VNC Viewer



ubuntu MATE



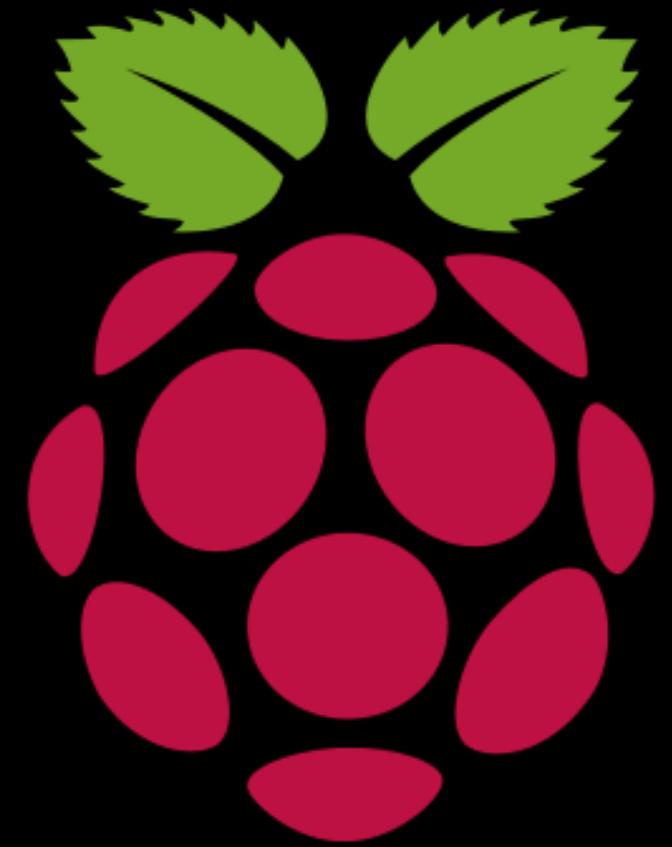
Windows IoT



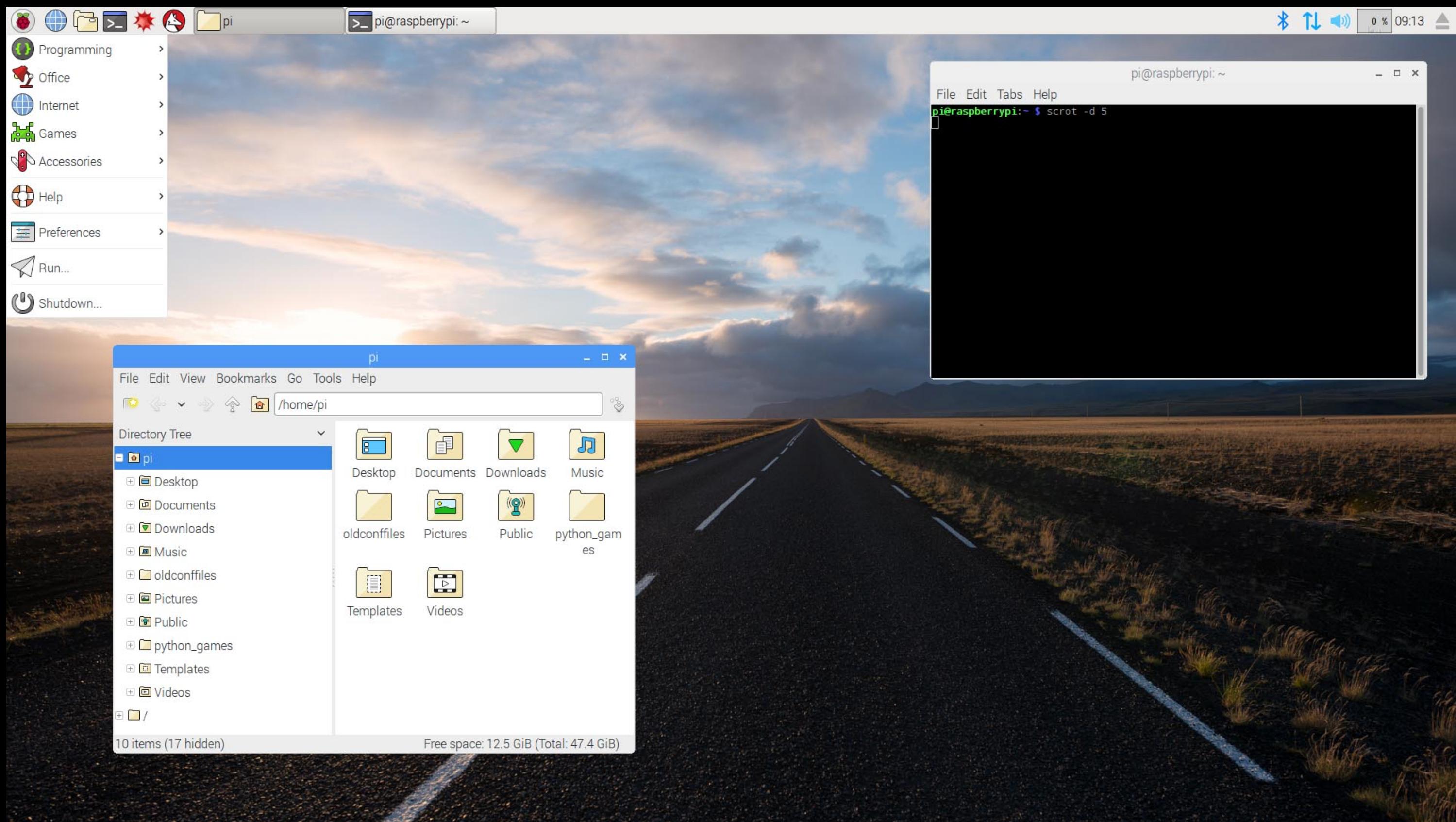
FLINT OS

Exemplos de Sistemas Operacionais para Raspberry Pi

# Raspbian (Jessie)



Sistema Operacional da Disciplina: Raspbian



Exemplo de Tela no Raspbian

Th

Editor de Código Python: Thonny

The screenshot shows the Thonny Python IDE interface. At the top, there's a toolbar with various icons. Below it is a menu bar with File, Edit, View, Run, Tools, and Help. The main area has tabs for 'hello.py' and 'countdown.py'. The 'countdown.py' tab is active, showing the following Python code:

```
n = 10

while n > 0:
    print(n)
    n-=1

print("Blast Off!")
```

Two specific icons in the toolbar are circled in red: a play button and a stop button. The word 'editor de texto' is written in red text on the right side of the code editor area.

In the bottom panel, there's a 'Shell' tab displaying the output of the code execution:

```
10
9
8
7
6
5
4
3
2
1
Blast Off!
>>>
```

The word 'tela de comandos' is written in red text on the right side of the shell output area.

Editor de Código Python: Thonny

# Revisão de Python



Linguagem de Programação Python

pt.wikipedia.org/wiki/Python

Python – Wikipédia, a enciclopédia livre

Não autenticado Discussão Contribuições Criar uma conta Entrar

Artigo Discussão Ler Editar Editar código-fonte Ver histórico Pesquisar na Wikipédia

# Python

Origem: Wikipédia, a enciclopédia livre.

**Nota:** Para outros significados, veja [Python \(desambiguação\)](#).

**Python** é uma linguagem de programação de alto nível,<sup>[4]</sup> interpretada, de script, imperativa, orientada a objetos, funcional, de tipagem dinâmica e forte. Foi lançada por Guido van Rossum em 1991.<sup>[1]</sup> Atualmente possui um modelo de desenvolvimento comunitário, aberto e gerenciado pela organização sem fins lucrativos Python Software Foundation. Apesar de várias partes da linguagem possuírem padrões e especificações formais, a linguagem como um todo não é formalmente especificada. O padrão *de facto* é a implementação CPython.

A linguagem foi projetada com a filosofia de enfatizar a importância do esforço do programador sobre o esforço computacional. Prioriza a legibilidade do código sobre a velocidade ou expressividade. Combina uma sintaxe concisa e clara com os recursos poderosos de sua biblioteca padrão e por módulos e frameworks desenvolvidos por terceiros.

Python é uma linguagem de propósito geral de alto nível, multi paradigma, suporta o paradigma orientado a objetos, imperativo, funcional e procedural. Possui tipagem dinâmica e uma de suas principais características é permitir a fácil leitura do código e exigir poucas linhas de código se comparado ao mesmo programa em outras linguagens. Devido às suas características, ela é principalmente utilizada para processamento de textos, dados científicos e criação de CGIs para páginas dinâmicas para a web.

O nome **Python** teve a sua origem no grupo humorístico britânico Monty Python,<sup>[5]</sup> criador do programa *Monty Python's Flying Circus*, embora muitas pessoas façam associação com o réptil do mesmo nome (em português, *pítão* ou *pitão*).

**Paradigma:** Multiparadigma: Orientação a objetos Programação imperativa Programação funcional

**Surgido em:** 1991 (26–27 anos)<sup>[1]</sup>

**Última versão:** 3.6.4 (19 de dezembro de 2017; há 11 meses)<sup>[2]</sup>

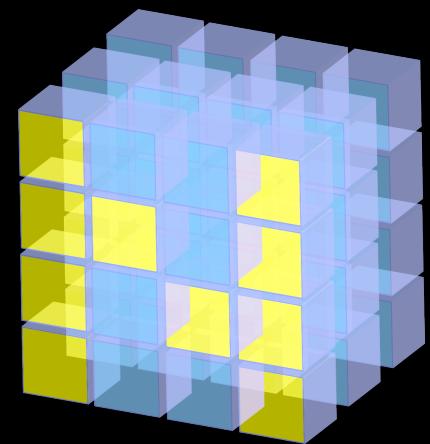
**Criado por:** Guido van Rossum<sup>[1]</sup>

**Estilo de tipagem:** Dinâmica, forte

**Dialetos:** CPython, IronPython, Jython, PyPy

**Influenciada por:** ABC,<sup>[3]</sup> ALGOL 68,<sup>[3]</sup> C<sup>[3]</sup>, Haskell, Icon, Java, Lisp,<sup>[3]</sup>

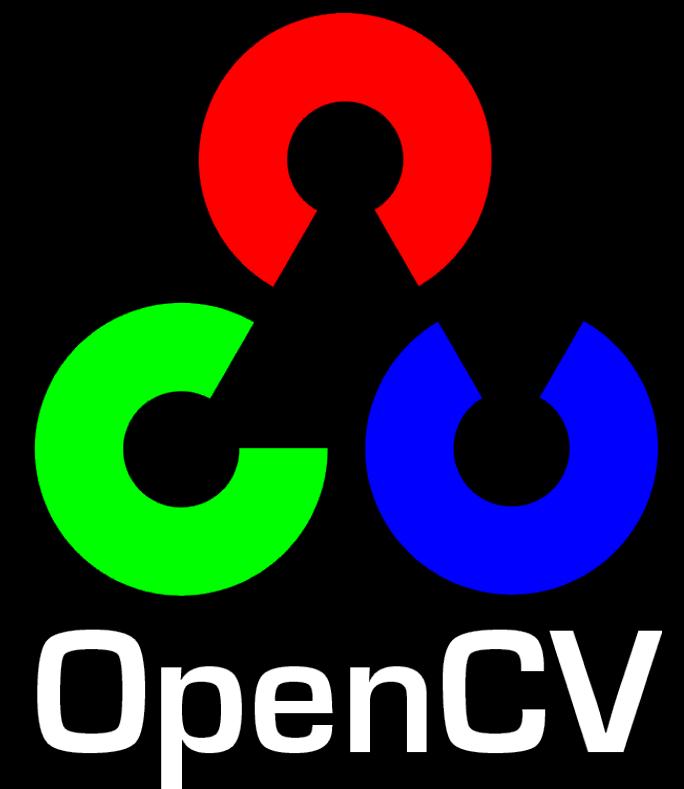
Origem do Python



NumPy



SciPy



Exemplos de Aplicações de Python

```
# código Python para verificar número primo
numero = int(input("Digite um número: "))

if numero > 1:
    for i in range(2, numero):
        if (numero % i) == 0:
            print(numero, "não é primo")
            break
    else:
        print(numero, "é primo")
else:
    print(numero, "não é primo")
```

Exemplo de Código em Python

```
# código Python para verificar número primo
numero = int(input("Digite um número:"))

if numero > 1:
    for i in range(2, numero):
        if (numero % i) == 0:
            print(numero, "não é primo")
            break
    else:
        print(numero, "é primo")
else:
    print(numero, "não é primo")
```

```
// código C para verificar número primo
#include <stdio.h>

int main ()
{
    int numero, i;
    printf("Digite um número: ");
    scanf("%d", &numero);

    if (numero > 1) {
        for (i = 2; i < numero; i++) {
            if (numero % i == 0) {
                printf("%d não é primo", numero);
                break;
            }
        }
        if (i == numero) {
            printf("%d é primo", numero);
        }
    } else {
        printf("%d não é primo", numero);
    }
}
```

```
>>> x = 2  
  
>>> x  
2  
  
>>> y = x * 4 + 10  
  
>>> y  
18  
  
>>> y + 1  
19  
  
>>> y  
18  
  
>>> y = y + 1  
  
>>> y  
19  
  
>>> variavel_com_nome_grande = (x - y) / 8
```

```
>>> lista = [1, 2, 3]
>>> texto = "Texto"
>>> texto2 = 'Texto'      # tanto faz usar " " ou ' '
>>> booleano = True
>>> booleano = False
>>> vazio = None
>>> funcao = print
>>> dicionario = {"campo 1": 2, "campo 2": 4}
```

*comentário explicativo com #*

```
>>> print("Olá!")
Olá!

>>> print("Valor:", 2)
Valor: 2

>>> sleep(4)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'sleep' is not defined

# importa função "sleep" da biblioteca "time"
>>> from time import sleep

# pausa programa por 4 segundos
>>> sleep(4)

# importa todas as funções da biblioteca "time"
>>> from time import *
```

```
>>> lista = [10, 20, 30, 40, 50]
>>> len(lista) # total de elementos (length)
5
>>> lista2 = ["abc", [0, 0], None] ← elementos de tipos diferentes
>>> lista + lista2 # concatenação de listas
[10, 20, 30, 40, 50, 'abc', [0, 0], None]
```

```
>>> lista = [10, 20, 30, 40, 50]
>>> lista[0]      # acesso ao primeiro elemento (índice 0)
10
>>> lista[1]      # acesso ao segundo elemento (índice 1)
20
>>> lista[4]      # acesso ao quinto elemento (índice 4)
50
>>> lista[5]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
IndexError: list index out of range
>>> lista[-1] / lista[-2] # último e penúltimo elemento
1.25
>>> lista[1:4] # elementos de índice 1 até ANTES de 4
[20, 30, 40]
```

```
>>> lista = [10, 20, 30, 40, 50]
>>> lista[0] = 15      # modifica o primeiro elemento
>>> lista.append(7)  # inclui elemento 7 no final
>>> lista
[15, 20, 30, 40, 50, 7]
>>> del lista[2]      # remove elemento no índice 2
>>> lista
[15, 20, 40, 50, 7]
>>> lista.sort()      # ordena lista
>>> lista
[7, 15, 20, 40, 50]
>>> lista.sort(reverse=True) # ordem decrescente
>>> lista
[50, 40, 20, 15, 7]
```

Lista

x

Dicionário

Índice



Elemento

0

1810001

Chave



Valor

1

9.5

"CR"

9.5

2

"JC"

"nome"

"JC"

3

[25, 12, 0]

"nascimento"

[25, 12, 0]

4

None

"pai"

None

...

...

...

...

```
>>> dados = {"nome": "Jan K. S.", "idade": 31}

# acessa valor a partir da chave
>>> dados["nome"]
'Jan K. S.'

# altera valor da chave
>>> dados["idade"] = 34

# adiciona chave/valor
>>> dados["sexo"] = "masculino"

# deleta chave/valor
>>> del dados["nome"]

>>> dados
{'idade': 34, 'sexo': 'masculino'}
```

```
>>> texto = "This is Sparta!"  
>>> texto = 'This is Sparta!'  
>>> len(texto)      # total de caracteres (length)  
15  
>>> texto[2]        # caracter no índice 2 = 3º caracter  
i  
>>> texto[0:4]       # caracteres de 0 até antes de 4  
This  
>>> texto.replace("is", "iiiiisssss")  
>>> texto  
Thiiiiisssss iiiiisssss Sparta!  
>>> "Madness? " + texto  
Madness? Thiiiiisssss iiiiisssss Sparta!  
>>> "Madness? " + "\n" + texto # "\n" = quebra de linha  
Madness?  
Thiiiiisssss iiiiisssss Sparta!
```

```
>>> numero = 100 / 15
>>> texto = "Resultado = "
>>> texto + numero
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: must be str, not float
>>> texto + str(numero)
'Resultado = 6.66666666666667'
>>> "Resultado (inteiro) = %d" % numero
'Resultado (inteiro) = 6'
>>> "Resultados = %.2f e %.4f" % (numero, numero/3)
'Resultados = 6.67 e 2.2222'
```

The diagram illustrates the conversion of a float to an integer and then to a string. It consists of three red curved arrows pointing from the right side of the first two code lines to the corresponding parts of the third line. The first arrow points from the placeholder %d in the string "Resultado (inteiro) = %d" to the variable numero. The second arrow points from the placeholder %.2f in the string "Resultados = %.2f e %.4f" to the variable numero. The third arrow points from the placeholder %.4f in the same string to the expression numero/3.

```
>>> texto_com_inteiro = "84"
>>> texto_com_inteiro / 2
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: unsupported operand type(s) for -: 'str' and
'int'
>>> int(texto_com_inteiro) / 2
42

>>> texto_com_decimal = "-4.55"
>>> texto_com_decimal * 2
'-4.55-4.55'
>>> float(texto_com_decimal) * 2
-9.1
```

```
>>> x = 2  
  
>>> x == 2 # x é igual a 2?  
True  
  
>>> x != 2 # x é diferente de 2?  
False  
  
>>> x > 3  
False  
  
>>> not x <= 0 # x não é menor ou igual a 0?  
True  
  
>>> x > 0 and x < 1 # x é maior que 0 E menor que 1?  
False  
  
>>> x <= -2 or x >= 2  
True  
  
>>> x in [1, 3, 5, 7] # x está na lista?  
False  
  
>>> "abc" in {"abc": 1, "cde": 10} # abc é uma chave?  
True
```

```
>>> x = 2                                não esqueça do ":" no final!  
    ↗  
>>> if x < 0:                          # se + verificação  
...     y = 4                            # indentação obrigatória!  
... elif x > 0 and x < 7: # caso contrário, se (else if)  
...     y = 3  
... else:                                # caso contrário  
...     y = 0  
  
...  
>>> y  
3
```

```
>>> lista = [21, 22, 23, 24]
# repete comandos para cada elemento da lista
>>> for elemento in lista:
...     print( elemento * 2 )
...
42
44
46
48
>>> for i in range(21, 24): # de 21 até ANTES de 24
...     print(i)
...     sleep(1)    # espera 1 segundo entre cada repetição
...
21
22
23
```

```
>>> x = 15

>>> while x > 1: # repita enquanto x for maior que 1
...     x = x / 5
...
...
...
>>> x
0.6

>>> while True: # repita PARA SEMPRE
...     print('Hey, listen!')
...     sleep(0.5)
...
...
...
Hey, listen!
Hey, listen!
Hey, listen!
...
...
```

```
>>> def faz_conta(x, y):  
...     return (x + y) / (x * y)  
...  
>>> def imprime_resultado(x):  
...     print("O resultado obtido foi: ", x)  
...  
>>> faz_conta(2, 4)  
0.75  
>>> imprime_resultado( faz_conta(2, 4) )  
O resultado obtido foi: 0.75
```

```
>>> x = 2

>>> def retorna_x_mais_1():
...     return x + 1
...
...
>>> def aumenta_x():
...     x = x + 2
...     return x
...
...
>>> retorna_x_mais_1()
3
>>> aumenta_x()
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
    File "<stdin>", line 2, in aumenta_x
      UnboundLocalError: local variable 'x' referenced
      before assignment
```



```
>>> global x
>>> x = 2
>>> def retorna_x_mais_1():
...     return x + 1
...
...
>>> def aumenta_x():
...     global x # necessário para alterar uma variável global
...     x = x + 2
...     return x
...
...
>>> aumenta_x()
4
>>> aumenta_x()
6
>>> aumenta_x()
8
```

# Resumo da Ópera

Funcionalidade	Comandos
Funções	<pre>x = input("Digite um número: ") • print("Resultado: ", x) from time import sleep • sleep(0.5)</pre>
Listas <a href="#">acessar documentação</a>	<pre>lista = [1, 2, 3] • lista2 = ["texto", [0, 0], 5] lista[0] • lista[1:3] • total_de_elementos = len(lista) lista.append(novo_elemento) • del lista[indice]</pre>
Dicionários <a href="#">acessar documentação</a>	<pre>dicionario = {"chave 1": 42, "chave 2": [1, 2, 3]} dicionario["chave 1"] • dicionario["chave 3"] = "Olá!"</pre>
Textos (Strings) <a href="#">acessar documentação</a>	<pre>texto = "olá!" • texto[0] • texto[1:4] • len(texto) texto + str(numero) • "x = %.2f, y = %d" % (num1, num2) 2 + int("11") • 4 / float("23.5")</pre>
Condicionais	<pre>if x != 0:     y = 4 elif x &gt;= 0:     y = 3 else:     y = 0</pre>
Repetições	<pre>for elem in lista:     ... for i in range(1, 4):     while x &gt; 1:         ... def funcao1(x):     return x + 2 def funcao2(x, y, z):     global x     x = 2</pre>
Criação de Funções e Variáveis Globais	