

# Introdução a programação Python

Fernando Fernandes dos Santos



# Info

- Dúvidas sobre o minicurso:  
fernandofernandesantos@gmail.com
- <https://bit.ly/2VOZqI3>
- Cronograma
  - Parte I
  - Intervalo
  - Parte II
  - Intervalo
  - Parte III
- Senhas máquinas: qwerty0

# Cronograma

- Parte I:
  - Introdução
  - Visão geral
  - Configuração do ambiente
  - Sintaxe básica
  - Tipos de variáveis
  - Operadores básicos
  - Operadores condicionais
  - Laços de repetição

# Cronograma

- Parte II:
  - Data e tempo
  - Funções
  - Modularidade
  - Arquivos
  - Tratamento de exceções
  - Orientação a objetos
  - Manipulação de arquivo CSV

# Cronograma

- Parte III:
  - Matplotlib e Pandas
  - Exemplo gráfico de pontos
  - Exemplo gráfico de barras
  - Exemplo gráfico de linhas
  - Exemplo de histograma
  - Exemplo de gráfico de grupos

# Cronograma

- **Parte I:**
  - **Introdução**
  - Visão geral
  - Configuração do ambiente
  - Sintaxe básica
  - Tipos de variáveis
  - Operadores básicos
  - Operadores condicionais
  - Laços de repetição

# Porque aprender Python

- Python é uma linguagem de programação criada em 1990
- Criador: Guido van Rossum
- Por quê???

```
#include <iostream>

int main(){
    ...
    for(int i = 1; i < 10; i++)
    {
        std::cout << "Iteration: " << i << "\n";
        for(int j = i; j < 100; j++){ if(j % i == 0) std::cout << i << "\n"; }
    }
    return 0;
}
```

# Porque aprender Python

- Python é uma linguagem de programação criada em 1990
- Criador: Guido van Rossum
- Por quê???

```
void tprintf(const char* format) // base function
{
    std::cout << format;
}

template<typename T, typename... Targs>
void tprintf(const char* format, T value, Targs... Fargs) // recursive variadic function
{
    for ( ; *format != '\0'; format++ ) {
        if ( *format == '%' ) {
            std::cout << value;
            tprintf(format+1, Fargs...); // recursive call
            return;
        }
        std::cout << *format;
    }
}

int main()
{
    tprintf("% world% %\n", "Hello", '!', 123);
    return 0;
}
```



# Porque aprender Python

- Python é uma linguagem de programação criada em 1990
- Criador: Guido van Rossum
- Por quê???
  - Para ser bonito e fácil de ler
  - Explícito é melhor que implícito
  - Simples é melhor do que complexo
  - Legibilidade importa muito

# Porque aprender Python

- Python é uma linguagem de programação criada em 1990
- Criador: Guido van Rossum
- Por quê???

C++

```
1 #include <iostream>
2
3 int main(){
4     std::cout << "Hello world\n";
5     return 0;
6 }
7
```

Python

```
1 print("Hello world")
2
3
```

# Porque aprender Python

- Python é uma linguagem de programação criada em 1990
- Criador: Guido van Rossum
- Por quê???

C++

```
fernando@fernando:~$ g++ helloworld.cpp -o helloworld
Hello world

real    0m0.004s
user    0m0.004s
sys     0m0.000s
fernando@fernando:~$
```

Python

```
fernando@fernando:~$ python helloworld.py
Hello world

real    0m0.024s
user    0m0.024s
sys     0m0.000s
fernando@fernando:~$
```

# Cronograma

- **Parte I:**

- Introdução
- **Visão geral**
- Configuração do ambiente
- Sintaxe básica
- Tipos de variáveis
- Operadores básicos
- Operadores condicionais
- Laços de repetição

# Visão geral

- Python é uma linguagem interpretada pelo interpretador Python em tempo de execução

# Visão geral

- Python é uma linguagem interpretada pelo interpretador Python em tempo de execução
- O programador pode interagir com o interpretador

# Visão geral

- Python é uma linguagem interpretada pelo interpretador Python em tempo de execução
- O programador pode interagir com o interpretador
- Python é uma linguagem orientada a objetos

# Visão geral

- Python é uma linguagem interpretada pelo interpretador Python em tempo de execução
- O programador pode interagir com o interpretador
- Python é uma linguagem orientada a objetos
- Python é uma linguagem “fácil” para se aprender programação



# Visão geral

- Python é uma linguagem interpretada pelo interpretador Python em tempo de execução
- O programador pode interagir com o interpretador
- Python é uma linguagem orientada a objetos
- Python é uma linguagem “fácil” para se aprender programação
- Portável e extensível

# Cronograma

- **Parte I:**

- Introdução
- Visão geral
- **Configuração do ambiente**
- Sintaxe básica
- Tipos de variáveis
- Operadores básicos
- Operadores condicionais
- Laços de repetição

# Configuração do ambiente

- A instalação do Python depende do sistema operacional em que será usado
  - [www.python.org/downloads](http://www.python.org/downloads)
- Variáveis de ambiente:
  - PYTHONPATH
  - PYTHONSTARTUP
- Executar script python  
`python <arquivo fonte>.py`

# Cronograma

- **Parte I:**

- Introdução
- Visão geral
- Configuração do ambiente
- **Sintaxe básica**
- Tipos de variáveis
- Operadores básicos
- Operadores condicionais
- Laços de repetição

# Sintaxe básica

- Função print
- Função input
- hello\_world.py

# Palavras reservadas

<b>and</b>	<b>assert</b>	<b>in</b>
<b>del</b>	<b>else</b>	<b>raise</b>
<b>from</b>	<b>if</b>	<b>continue</b>
<b>not</b>	<b>pass</b>	<b>finally</b>
<b>while</b>	<b>yield</b>	<b>is</b>
<b>as</b>	<b>break</b>	<b>return</b>
<b>elif</b>	<b>except</b>	<b>def</b>
<b>global</b>	<b>import</b>	<b>for</b>
<b>or</b>	<b>print</b>	<b>lambda</b>
<b>with</b>	<b>class</b>	<b>try</b>
<b>exec</b>		

# Indentação e comentários

- Toda a indentação em Python é feita pelo número de espaços
- Linhas múltiplas separadas por \
- Aspas ' e "
- Comentários por # e '''

# Exercícios

- Crie um programa em Python que leia seu nome, sobrenome e escreva na tela
- “<seu nome> no minicurso de Python” usando a função print e a função input



# Cronograma

- **Parte I:**

- Introdução
- Visão geral
- Configuração do ambiente
- Sintaxe básica
- **Tipos de variáveis**
- Operadores básicos
- Operadores condicionais
- Laços de repetição

# Atribuindo valores a variáveis

- Muito parecido com outras linguagens de programação
  - $X = 10$
- Possui outras facilidades, como atribuição múltipla
  - $a, b, c, d = 10, 20, 30, 10$

# Números

- As variáveis que contém números são criadas quando é atribuído um valor para a variável
  - $X = 10$
  - $a = 2.3$
  - $c = 99$
- Para apagar a qualquer referência a variável
  - `del X`
  - `del a,c`

# Tipos de números

- Inteiros e inteiros longos
  - Exemplos: 3, 422, 338, 3939, 38383838383L
- Float (números reais)
  - Exemplos: 3.14, 67.44, 88.2
- Complexos
  - Possuem parte x e y: x real e y imaginária
  - `var1 = complex(4.0, 3.0)`
- Conversão de um tipo para outro
  - `int(x)`, `long(x)`, `float(x)`, `complex(x, y)`

# Funções matemáticas definidas

1.  $\text{abs}(x)$  – valor absoluto de  $x$
2.  $\text{max}(x_1, x_2, \dots, x_n)$  – máximo elemento da lista
3.  $\text{min}(x_1, x_2, \dots, x_n)$  – mínimo elemento da lista
4.  $\text{pow}(x, y)$  – o mesmo que  $x^{**}y$
5.  $\text{round}(x, n)$  – arredondamento de  $x$  no  $n$  dígitos

# Funções da biblioteca math

1. `math.sqrt(x)` – raiz quadrada de  $x$
2. `math.ceil(x)` – arredondamento para cima  $x$
3. `math.exp(x)` – exponencial de  $x$
4. `math.floor(x)` – truncamento de  $x$
5. `math.log(x)` – logaritmo natural de  $x$
6. `math.log10(x)` – logaritmo na base 10 de  $x$
7. Trigonométricas
  1. `math.sin(x)`, `math.cos(x)`, `math.tan(x)` etc –  $x$  em radianos

# Funções de números randômicos

1. `random.choice(sequencia)` – escolhe um elemento da sequencia, pode ser lista, tupla, ou string
2. `random.randrange(start, stop, step)` – escolhe um número do intervalo definido start até stop, com diferença de step
3. `random.random()` – retorna um float randômico entre 0 – 1
4. `random.uniform(start, stop)` – escolhe um intervalo definido de start até stop
5. `random.shuffle(sequencia)` – embaralha uma sequencia

# Exercícios

- Usar a função max e min para encontrar o maior e menor elemento da lista: 2.4857, -8.3, 0.02, 10, -1
- Arredonde os valores a seguir com 3 dígitos:
  - $\frac{23}{9}$
  - $\sqrt{34.5}$
  - $\frac{1}{3^4}$
  - $e^{\log_{10} 3.4}$
- Verifique o resultado de:
  - $\sin^2 23 + \cos^2 23$



# Strings

- Uma string é um conjunto de caracteres
  - `var1 = "Hello world!"`
  - `var2 = "Esse e o minicurso de Python"`
- Strings são imutáveis em Python
- Principais operadores:
  - `*`, `+`, `[]`, `[:]`, `%`, `{}`
- Strings longas
  - `string_longa = """essa e uma string longa que tem mais de uma linhas"""`

# Strings

- `capitalize()` – retorna uma cópia da string com a primeira letra em maiúsculo
- `isnumeric()` – retorna verdadeiro se toda a string é constituída de números
- `len(string)` – retorna o tamanho da string
- `lower()/upper()` – retorna uma cópia da string transformada toda em minúsculo/maiúsculo
- `split(delimitador)` – retorna uma lista de strings dividida pelo delimitador

# Exercícios

- Leia uma frase de entrada e mostre na tela a frase de entrada com todas as letras em maiúsculo.
- Leia uma frase de entrada e mostre na tela a frase de entrada com todas as letras em minúsculo.
- Leia seu nome completo e depois separe ele separado por espaço
- Mostre na tela quantas letras tem o seu nome

# Listas

- Estrutura de dados mais básica em Python
  - `lista1 = [2, 3, 2, 3, 4]`
  - `lista2 = ['minicurso', 2, 3, 2.3]`
- O acesso é feito pelo índice da lista:
  - `lista1[2]` é 2
- Atualizando ou apagando elementos
  - `lista1[2] = 99`
  - `del lista1[4]`

# Listas

- `append()` – adiciona elemento no final
- `count(x)` – conta quantas vezes o `x` acontece na lista
- `extend(lista1)` – concatena uma `lista1` na lista que chamou o método
- `insert(índice, x)` – insere `x` na posição `índice`
- `pop()` – remove o último elemento da lista e o retorna
- `sort()` – ordena a lista

# Exercícios

- Inicialize uma lista com os seguintes elementos
  - 2, 1, 3, 12, 4.2, 85, 'naodeveriaestaraqui', 9.123, 1, 2, 4.2, 1, 3, 85
- Conte quantas vezes 2 aparece na lista
- Adicione o número 3.14 no final da lista
- Insira “minicurso” na posição 3

# Tuplas

- Tuplas são uma sequência de elementos que não pode ser modificada. “Uma lista constante”
  - tupla1 = (2, 3, 2, 3, 4)
  - tupla2 = ('minicurso', 2, 'essa e uma tupla', 2.3)
- Principais métodos:
  - min – menor elemento da tupla
  - len – tamanho da tupla

# Exercícios

- Crie duas tuplas
  - `tupla1 = (2, 3, 2, 3, 4)`
  - `tupla2 = ('minicurso', 2, 'essa e uma tupla', 2.3)`
- Crie uma terceira tupla
  - `tupla3 = tupla1 + tupla2`
- Mostre na tela o tamanho da tupla3



# Dicionários

- São estruturas de dados onde o índice pode ser número, strings, ou tuplas
  - `dicionario1 = {'nome' : 'Fernando', 'idade': 28, 'endereco' : "rua xx, num 22"}`
- Acesso é feito através da chave que vem antes do :
- O dado é o que vem depois do :
- Dicionários podem armazenar qualquer tipo de dado

# Exercícios

- Crie um programa que peça 3 códigos de usuário
- Cada código de usuário será uma chave para um dicionário que armazenará um outro dicionário que contém os dados do usuário
- Os dados de cada usuário será lido da tela: nome e idade
- Exemplo
  - `dic_users = { 1 : {'nome': 'Fernando', idade: 28}...`
- Mostre na tela os dados de cada usuário

# Cronograma

- **Parte I:**

- Introdução
- Visão geral
- Configuração do ambiente
- Sintaxe básica
- Tipos de variáveis
- **Operadores básicos**
- Operadores condicionais
- Laços de repetição

# Operadores aritméticos

- Adição
  - $a + b$
- Subtração
  - $a - b$
- Multiplicação
  - $a * b$
- Divisão
  - $b / a$
- Resto da divisão
  - $b \% a$
- Expoente
  - $a ** b$  ( $a^b$ )
- Divisão com truncamento (9 dividido por 2 resulta em 4)
  - $a // b$

# Operadores combinados

- Adiciona e atribui
  - $a += b$  ( $a = a + b$ )
- Subtrai e atribui
  - $b -= a$  ( $b = b - a$ )
- Multiplica e atribui
  - $a *= b$  ( $a = a * b$ )
- Divide e atribui
  - $a /= b$  ( $a = a / b$ )
- Calcula o resto e atribui
  - $a \% = b$  ( $a = a \% b$ )
- Calcula a potência e atribui
  - $a ** b$  ( $a = a ** b$ )
- Divide, arredonda e atribui
  - $a // = b$  ( $a = a // b$ )

# Exercícios

- Dado o vetor
  - $\vec{v} = \{3, 4, 2, 5, 6\}$
- Calcule a norma do vetor sem usar as funções sqrt e pow
  - $||\vec{v}|| = \sqrt{x_1^2 + x_2^2 + \dots + x_n^2}$
- Calcule o vetor normalizado
  - $\hat{v} = \frac{\vec{v}}{||\vec{v}||}$

# Operadores lógicos

- Igualdade
  - $a == b$  (a é igual a b?)
- Diferença
  - $a != b$  (a é diferente de b?)
- Maior e menor
  - $a > b$  (maior que) e  $a < b$  (menor que)
- Maior/menor e igual
  - $a >= b$  e  $a <= b$

# Operadores lógicos

- Operadores and e or são operadores da lógica booleana.
- E
  - Só é verdade quando a e b são verdadeiros
  - a and b (a e b)
- OU
  - Só é verdade quando pelo menos a ou b é verdadeiro
  - a or b (a ou b)
- Negação
  - not a (nega o valor de a)



# Operadores está contido, é e não é

- Está contido
  - $a \in b$  (a está contido em b)
- Não está contido
  - $b \notin a$  (b não está contido em a)
- É
  - $a \text{ is } b$  (a aponta para o mesmo objeto que b)
- NÃO É
  - $a \text{ is not } b$  (a não aponta para o mesmo objeto que b)

# Exercícios

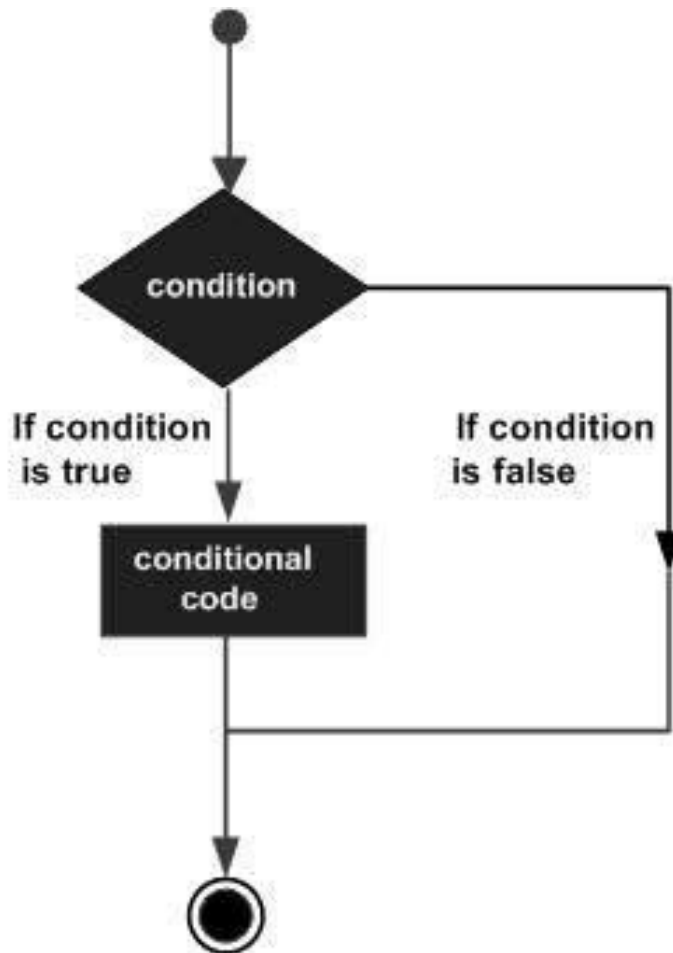
- Leia dois números da tela e mostre se eles são iguais ou não
- Leia duas strings da tela e mostre se a primeira string está contida na segunda

# Cronograma

- **Parte I:**

- Introdução
- Visão geral
- Configuração do ambiente
- Sintaxe básica
- Tipos de variáveis
- Operadores básicos
- **Operadores condicionais**
- Laços de repetição

# Operadores condicionais



```
if condição:  
    ...código...  
else:  
    ...código...
```

```
if condição:  
    ...código...  
elif nova_condiçao:  
    ...código...
```

# Exercícios

- Leia um número da tela e mostre se ele é divisível por 2
- Leia um nome completo da tela e mostre se este contém alguns dos seguintes nomes/sobrenomes
  - Enzo
  - João
  - José
  - Silva
  - Santos
  - Caso não contenha, escreva “Não contém”
- Leia 3 números e mostre o maior, o menor, e o do meio

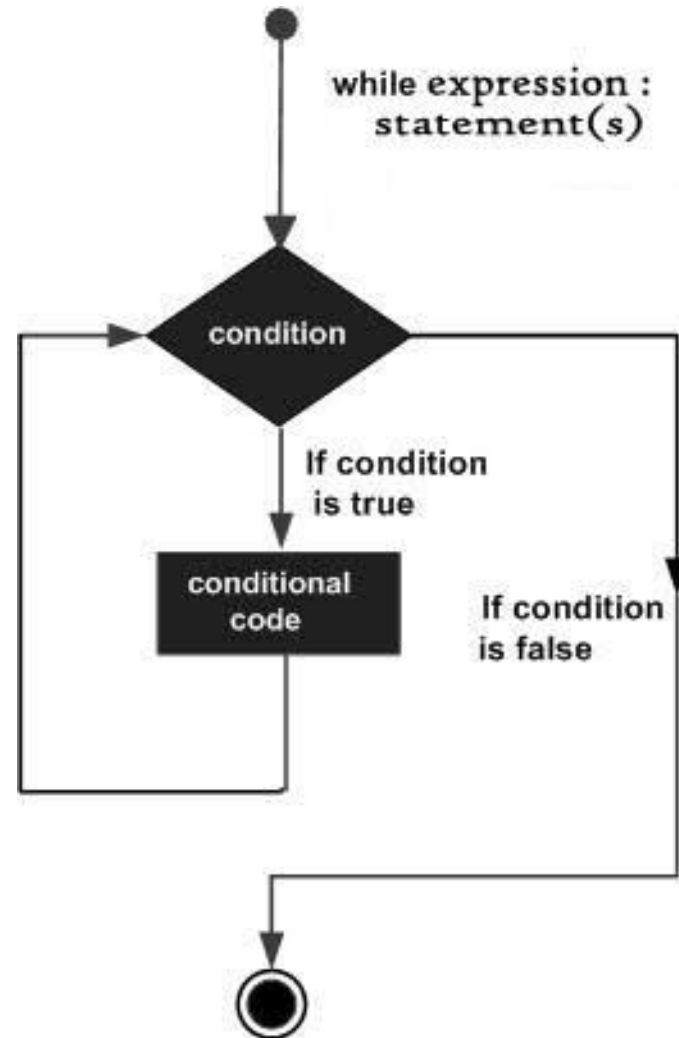
# Cronograma

- **Parte I:**

- Introdução
- Visão geral
- Configuração do ambiente
- Sintaxe básica
- Tipos de variáveis
- Operadores básicos
- Operadores condicionais
- **Laços de repetição**

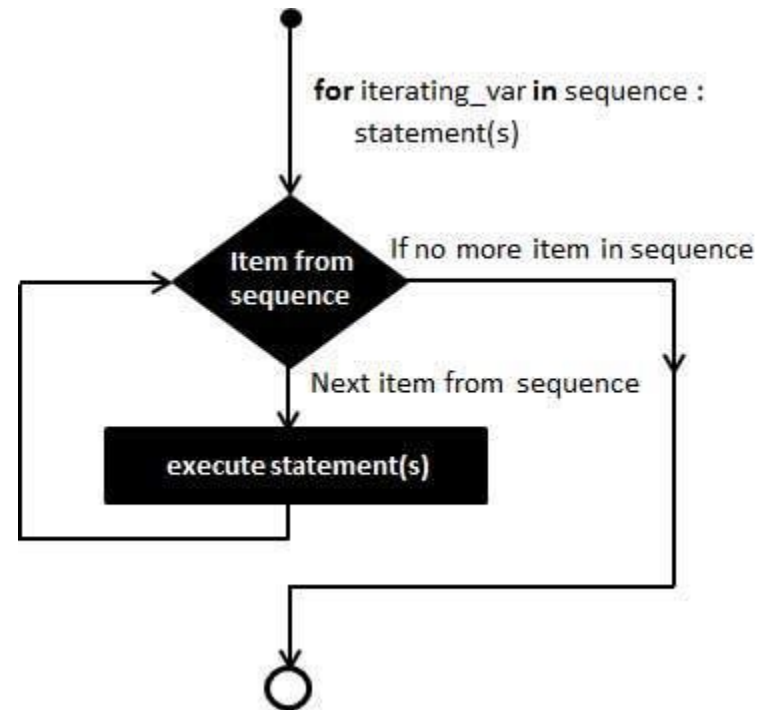
# While

while expressão:  
...código...



# For

for cada\_elemento in sequencia:  
...código...





# Exercício

- Ler um número da tela e definir se ele é primo ou não
- Crie uma lista de 100 posições com números randômicos de 0 a 1.
  - Calcule o somatório dessa lista
  - Crie uma segunda lista em que cada valor é o quadrado do primeira
  - Crie uma terceira lista em que cada valor é
$$\text{lista3} = \text{lista1} + \text{lista2}$$

Intervalo

# Cronograma

- Parte II:
  - Data e tempo
  - Funções
  - Modularidade
  - Arquivos
  - Manipulação de arquivo CSV
  - Tratamento de exceções
  - Orientação a objetos

# Cronograma

- Parte II:
  - Data e tempo
  - Funções
  - Modularidade
  - Arquivos
  - Manipulação de arquivo CSV
  - Tratamento de exceções
  - Orientação a objetos

# Bibliotecas time e datetime

- É possível manipular o tempo e datas de diferentes maneiras em Python
- As principais bibliotecas são
  - time
  - datetime
- `time.localtime(time.time())`
- `time.asctime(time.localtime(time.time()))`
- `time.clock()` – tempo em segundos em um instante
- `time.sleep(seconds)` – dorme por segundos

# Bibliotecas time e datetime

- datetime possui algumas funções mais elaboradas para manipulação de datas
- datetime(ano, mês, dia) – uma data específica
- date.today() – data de hoje do sistema operacional
- timedelta(days=d, seconds=s, microseconds=us, milliseconds=ms, minutes=m, hours=h, weeks=w)
- Timedelta é útil quando temos que comparar uma data com um intervalo de tempo

# Exercícios

- Calcular quantos dias se passaram desde a data de seu nascimento
- Calcular quantos dias se passaram desde o começo do ano

# Cronograma

- Parte II:
  - Data e tempo
  - Funções
  - Modularidade
  - Arquivos
  - Manipulação de arquivo CSV
  - Tratamento de exceções
  - Orientação a objetos



# Funções

- Funções são um bloco de instruções definidos e organizado de modo que possa ser reusado

```
def nome_da_funcao(parametros):  
    ...código...  
    return [expressão]
```

# Funções

- É possível definir argumentos padrões na chamada da função
  - `def funcao(arg1, arg2='Padrao'):`...

# Funções

- É possível definir argumentos padrões na chamada da função
  - `def funcao(arg1, arg2='Padrao'):...`
- Número de argumentos variáveis
  - `def funcao(arg1, *argumentos_variaveis)`

# Funções

- É possível definir argumentos padrões na chamada da função
  - `def funcao(arg1, arg2='Padrao'):...`
- Número de argumentos variáveis
  - `def funcao(arg1, *argumentos_variaveis)`
- Passar argumentos como “dicionário”
  - `def funcao(arg1, **dic_argumentos)`

# Funções

- É possível definir argumentos padrões na chamada da função
  - `def funcao(arg1, arg2='Padrao'):...`
- Número de argumentos variáveis
  - `def funcao(arg1, *argumentos_variaveis)`
- Passar argumentos como “dicionário”
  - `def funcao(arg1, **dic_argumentos)`
- Variáveis globais são acessíveis de todas as funções

# Funções

- É possível definir argumentos padrões na chamada da função
  - `def funcao(arg1, arg2='Padrao'):...`
- Número de argumentos variáveis
  - `def funcao(arg1, *argumentos_variaveis)`
- Passar argumentos como “dicionário”
  - `def funcao(arg1, **dic_argumentos)`
- Variáveis globais são acessíveis de todas as funções
- Variáveis locais são acessíveis somente na função

# Exercícios

- Faça uma função que receba uma string e defina se ela é palíndromo ou não. Exemplos
  - SATOR AREPO TENET OPERA ROTAS – é palíndromo
  - “O lavrador diligente conhece a rota do seu arado” – não é palíndromo
  - Arara – é palíndromo
- Faça uma função que receba o raio de um círculo e retorne a área
- Faça uma função que receba uma lista e retorne outra lista sem elementos duplicados

# Cronograma

- Parte II:
  - Data e tempo
  - Funções
  - **Modularidade**
  - Arquivos
  - Manipulação de arquivo CSV
  - Tratamento de exceções
  - Orientação a objetos



# Modularidade

- É possível reutilizar um conjunto de funções através de módulos
- Um módulo é um arquivo com funções relacionadas onde estas podem ser importadas em outro código fonte
- Python busca os módulos na seguinte ordem:
  - No diretório do código fonte
  - Nos diretórios contidos na variável de ambiente PYTHONPATH
  - Por último na pasta de instalação do Python

# Modularidade

- `dir()` – lista todos os módulos importados naquele arquivo
- Quando o projeto é separado em diferentes pastas
  - `MeuProjeto/`
    - `Pasta1/funcao_1.py`
    - `Pasta2/funcao_2.py`
    - `funcao_principal.py`
- Todos os diretórios tem que conter o arquivo `__init__.py`

# Exercícios

- Crie um módulo Python com as funções criadas nos exercícios anteriores e depois importe o módulo em outro arquivo fonte

# Cronograma

- **Parte II:**
  - Data e tempo
  - Funções
  - Modularidade
  - **Arquivos**
  - Manipulação de arquivo CSV
  - Tratamento de exceções
  - Orientação a objetos

# Abrindo e fechando arquivos

- Até agora nós usamos somente entrada e saída no terminal
- Para grande volume de dados é necessário utilizar arquivos
- Função open
  - `open(nome_do_arquivo, modo_leitura_escrita)`
- `nome_do_arquivo` exemplo: `./arquivo.csv`, `graduacao/tcc.doc`, ....
- `modo_leitura_escrita`: é como a função irá abrir o arquivo

# Modos de abertura

Leitura		Escrita		Escrita no final	
r	Abre um arquivo de texto	w	Abre para escrita, sobrescreve	a	Abre para escrita no final do arquivo
rb	Abre arquivo binário para leitura	wb	Abre arquivo binário, sobrescreve	ab	Abre arquivo binário para escrita no final
r+	Abre um arquivo de texto leitura e escrita	w+	Abre arquivo para leitura e escrita, sobrescreve	a+	Abre arquivo para leitura e escrita no final
rb+	Abre um arquivo binário para leitura escrita	wb+	Abre um arquivo binário para leitura escrita, sobrescreve	ab+	Abre arquivo binário para leitura e escrita no final

# Manipulando o arquivo

- Atributos de um arquivo
  - name – nome do arquivo
  - closed – identifica se o arquivo está fechado ou não
  - mode – modo no qual o arquivo foi aberto
- Para fechar um arquivo
  - `arquivo.close()`
- Para escrever uma string em um arquivo
  - `arquivo.write(string)`
- Para ler 10 bytes do arquivo
  - `arquivo.read(10)`
- Para ler uma linha do arquivo
  - `arquivo.readline()`

# Exercício

- Acesse a página da Wikipedia sobre a UFRGS em inglês
- Copie o primeiro paragrafo linha a linha e cole em um arquivo .txt
- Escreva um pequeno programa que abra esse arquivo e substitua todos os acrônimos UFRGS por uma frase de sua escolha



# Cronograma

- Parte II:
  - Data e tempo
  - Funções
  - Modularidade
  - Arquivos
  - Tratamento de exceções
  - Orientação a objetos
  - Manipulação de arquivo CSV

# Estrutura try-except

- Quando um erro inesperado acontece chamamos de exceções
- Python possui meios de tratar quando exceções acontecem

**try:**

**...código que devia executar sem erros...**

**except Error1:**

**... código que executa caso Error1 aconteça...**

**except Error2:**

**... código que executa caso Error2 aconteça...**

# Principais exceções

- Exception – Base de todas as exceções
- StopIteration – Quando está iterando sobre uma sequência e essa termina abruptamente
- SystemExit – Exceção invocada por chamada da função exit()
- ArithmeticError – Exceção causada por algum erro aritmético
- OverflowError – Quando estoura a representação de inteiros e floats
- ZeroDivision – Divisão por zero

# Exercícios

- Trate o erro que será causado por uma divisão por zero

# Cronograma

- Parte II:

- Data e tempo
- Funções
- Modularidade
- Arquivos
- Tratamento de exceções
- **Orientação a objetos**
- Manipulação de arquivo CSV

# Classes

- Classes em programação são um meio de representar objetos e fenômenos do mundo real em estrutura de dados

# Classes

- Classes em programação são um meio de representar objetos e fenômenos do mundo real em estrutura de dados
- Exemplo: Classe Trabalhador, classe Carro, classe Animal...

# Classes

- Classes em programação são um meio de representar objetos e fenômenos do mundo real em estrutura de dados
- Exemplo: Classe Trabalhador, classe Carro, classe Animal...
- Cada classe possui atributos que são dados contidos no objeto



# Classes

- Classes em programação são um meio de representar objetos e fenômenos do mundo real em estrutura de dados
- Exemplo: Classe Trabalhador, classe Carro, classe Animal...
- Cada classe possui atributos que são dados contidos no objeto
- Exemplo: Trabalhador.nome, Carro.velocidade\_max, Animal.e\_domestico

# Classes

- Cada classe possui métodos, que são funções que definem algum comportamento da classe
- Exemplo: `Trabalhador.imprime_nome`

```
class Trabalhador:  
    def __init__(self, nome, salario):  
        self.salario = salario  
        self.nome = nome  
  
    def imprime_nome(self):  
        print(self.nome)  
  
    def imprime_salario(self):  
        print(self.salario)
```

# Exercício

- Defina uma classe chamada Estudante
- O método `__init__` irá receber
  - Nome, Idade, semestre, e uma lista contendo todas as cadeira a qual ele está cursando
- Implementar um método `imprime_informacao()` onde este irá imprimir o Nome, idade, semestre, e as cadeiras que esse estudante está cursando

# Cronograma

- Parte II:
  - Data e tempo
  - Funções
  - Modularidade
  - Arquivos
  - Tratamento de exceções
  - Orientação a objetos
  - Manipulação de arquivo CSV

# Arquivos CSV

- CSV – Comma Separated Values
  - Alguns arquivos CSV podem usar ; ou outros caracteres para separar os valores
- Um arquivo CSV pode ser aberto por programas tipo Excel e Libreoffice Calc

Cabeçalho	→	coluna1, coluna2,coluna3,coluna4,coluna5,coluna6
Dados	{	0.01214,0.07936,0.07799,0.07994,0.03393,0.02225
		0.05841,0.0011,0.05875,0.06374,0.07521,0.09295
		0.09843,0.00169,0.02203,0.02879,0.04721,0.07074
		0.06297,0.08871,0.01439,0.08867,0.09371,0.02379
		....

# DictReader e DictWriter

- Para manipular arquivos .csv é necessário importar a biblioteca csv
- DictReader – Objeto contém as funções para ler o arquivo CSV
- DictWriter – Objeto contém as funções para escrever no arquivo CSV
  - É necessário definir o cabeçalho antes de criar o objeto

# Exercício

- Baixe o arquivo `deter_amz_aviso.csv` do Github do minicurso
  - Dados do sistema DETER do INPE
  - Avisos de desmatamento da Amazônia
- Abrir o arquivo `deter_amz_aviso.csv` e importar os dados em uma lista
- Encontrar os diferentes entes federativos que possuem avisos de desmatamento informados pelo sistema DETER do INPE.

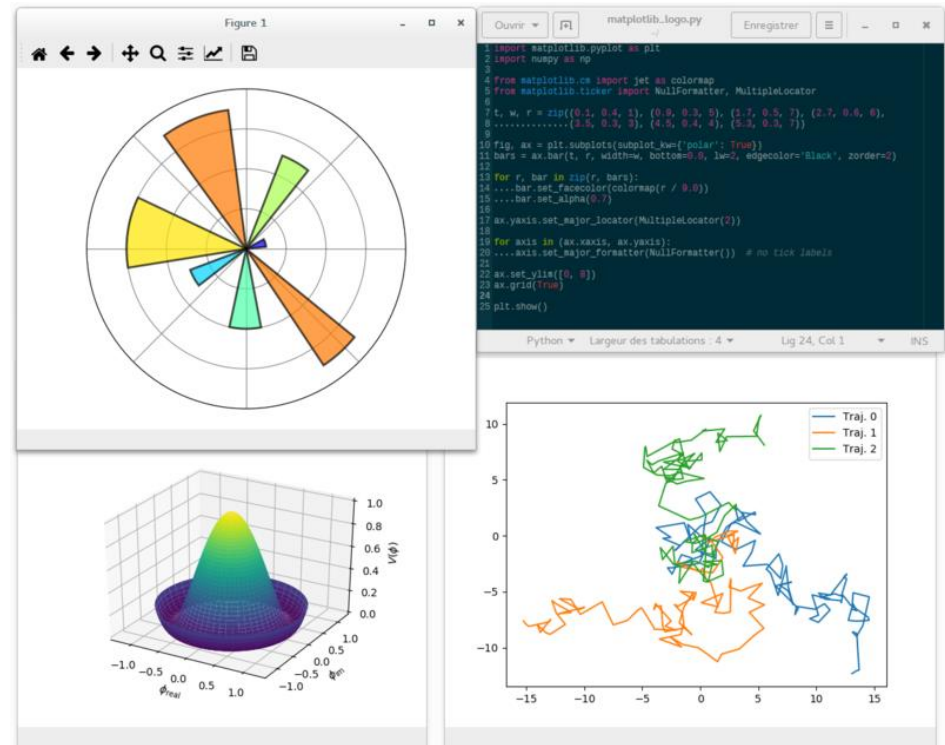
# Cronograma

- Parte III:
  - Matplotlib e Pandas
  - Exemplo gráfico de pontos
  - Exemplo gráfico de barras
  - Exemplo gráfico de linhas
  - Exemplo de histograma
  - Exemplo de gráfico de grupos



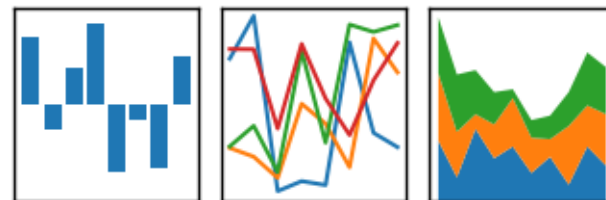
# Matplotlib e Pandas

- Matplotlib é uma biblioteca de criação de gráficos no Python
- Pandas é uma biblioteca de manipulação de dados



pandas

$$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$$



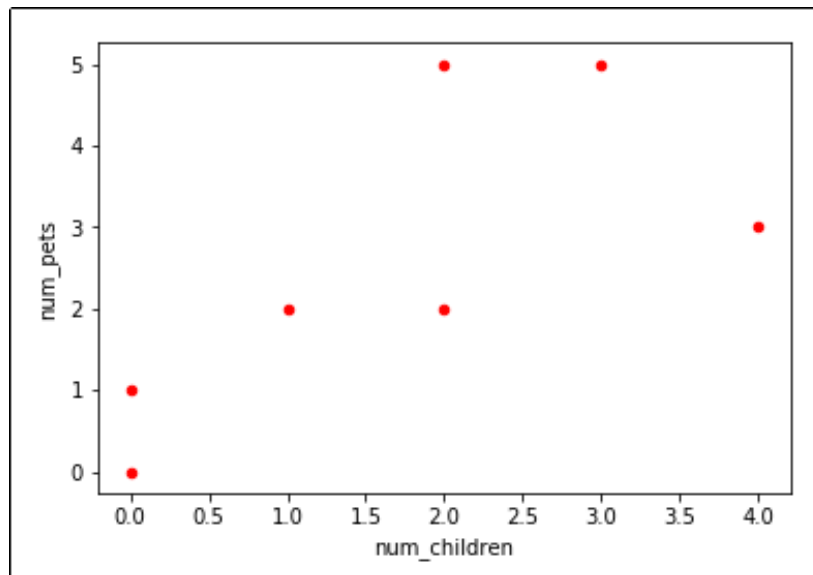
# Exemplo base

```
{  
'name':      ['john','mary','peter','jeff','bill','lisa','jose'],  
'age':       [23,78,22,19,45,33,20],  
'gender':    ['M','F','M','M','M','F','M'],  
'state' :    ['california','dc','california','dc','california','texas','texas'],  
'num_children': [2,0,0,3,2,1,4],  
'num_pets':   [5,1,0,5,2,2,3]  
}
```

	name	age	gender	state	num_children	num_pets
0	john	23	M	california	2	5
1	mary	78	F	dc	0	1
2	peter	22	M	california	0	0
3	jeff	19	M	dc	3	5
4	bill	45	M	california	2	2
5	lisa	33	F	texas	1	2
6	jose	20	M	texas	4	3

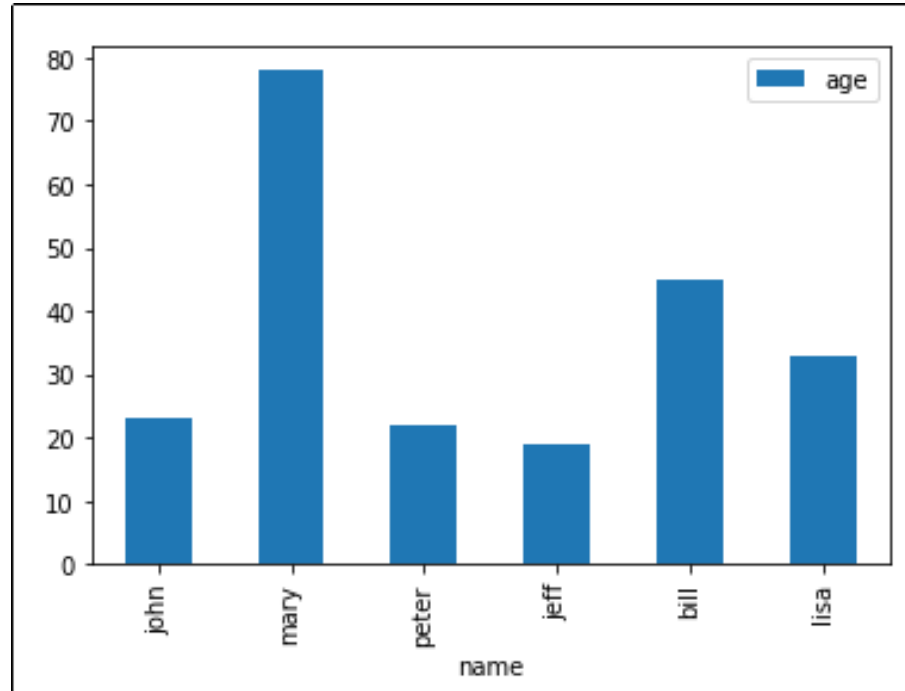
# Gráfico de pontos

- Gerar um gráfico de pontos com num\_children no eixo X, e num\_pets no eixo Y
- `df.plot(kind='scatter', x='num_children', y='num_pets', color='red')`



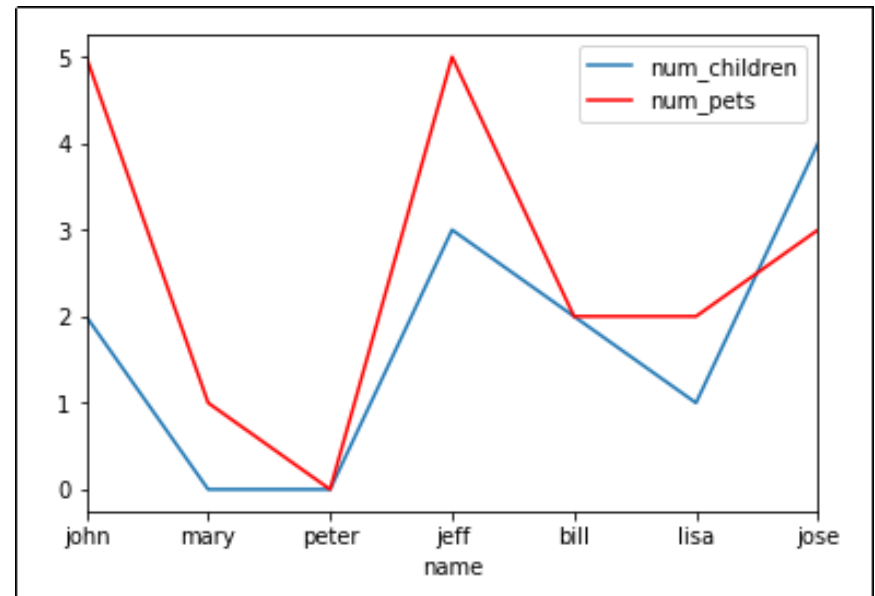
# Gráfico de barras

- Gerar um gráfico com age por name
- `df.plot(kind='bar', x='name', y='age')`



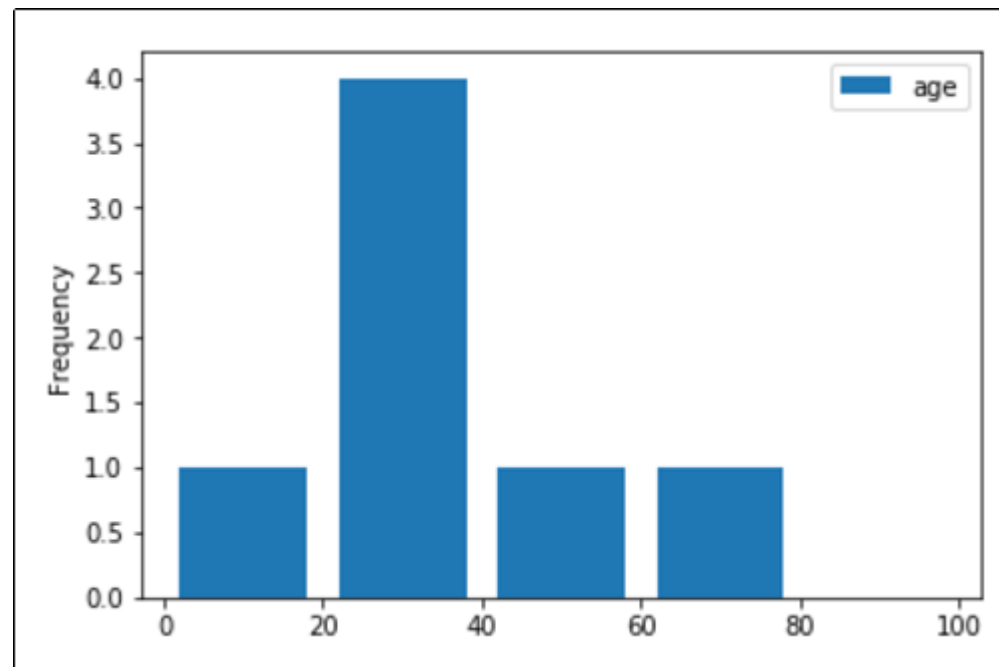
# Gráfico de linhas

- Quando é necessário desenhar mais de uma variável no mesmo eixo temos que usar a função `ax = plt.gca()`
- Eixo x: name
- Eixo y: num\_children e num\_pets



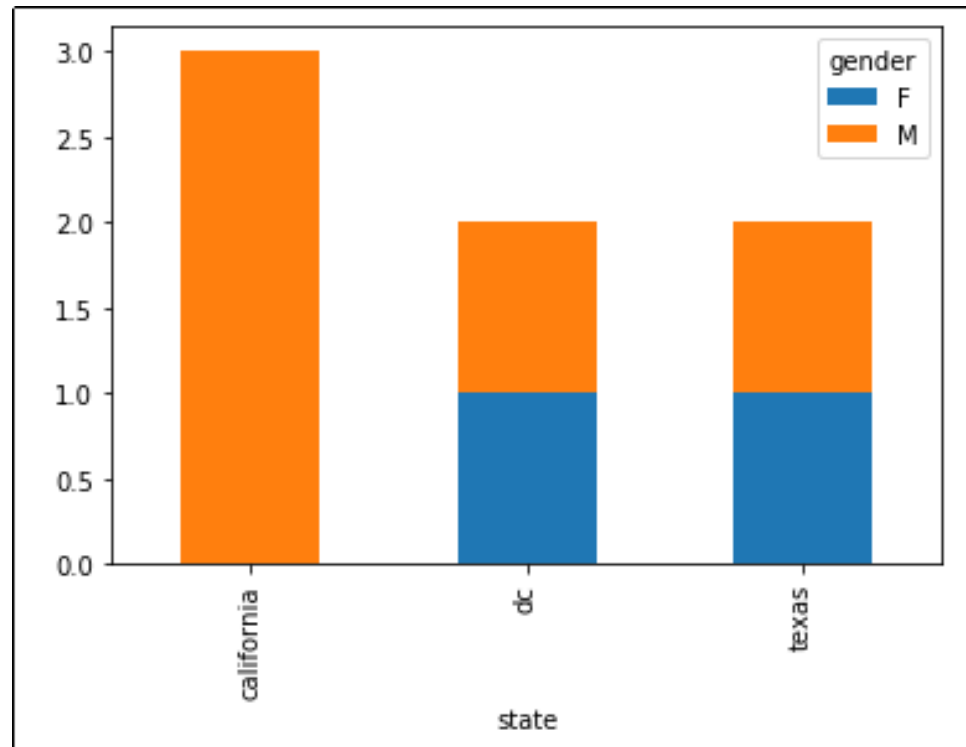
# Histograma

- Para criar um histograma é necessário selecionar uma variável do DataFrame
- `df[['age']].plot(kind='hist', bins=[20,40,60,80,100], rwidth=0.8)`



# Gráfico de grupos

- `df.groupby(['state', 'gender']).size().unstack().plot(kind='bar', stacked=True)`



# Exercício

- Faça um programa que abra o arquivo com os avisos de desmatamento na Amazônia e transforme os dados em um dicionário parecido com o que foi mostrado no exemplo anterior
- Plote um gráfico de grupos para os estados que tiveram aviso



É isso

- Dúvidas sobre o minicurso:  
fernandofernandesantos@gmail.com
- Material no Github