

```

class Network(object):

    def __init__(self, sizes):
        self.num_layers = len(sizes)
        self.sizes = sizes
        self.biases = [np.random.randn(y, 1) for y in sizes[1:]]
        self.weights = [np.random.randn(y, x) for x, y in zip(sizes[:-1], sizes[1:])]

    def feedforward(self, a):
        for b, w in zip(self.biases, self.weights):
            a = sigmoid(np.dot(w, a)+b)
        return a

    def SGD(self, training_data, epochs, mini_batch_size,
            training_function, test_data=None):
        """Train the neural network using mini-batch stochastic gradient descent.

        Args:
            training_data: list of tuples (x, y) where x is a vector
                           of inputs and y is a vector of target outputs
            epochs: number of epochs (full passes over the training data)
            mini_batch_size: number of training data points to use in each mini-batch
            training_function: function that takes a list of inputs and a list of targets
                               and returns a list of gradients
            test_data: list of tuples (x, y) where x is a vector of inputs
                       and y is a vector of target outputs

        Returns:
            The final state of the network after training.
        """
        n = len(training_data)

        for j in range(epochs):
            random.shuffle(training_data)
            mini_batches = [training_data[k:k+mini_batch_size]
                             for k in range(0, n, mini_batch_size)]
            for mini_batch in mini_batches:
                self.update_mini_batch(mini_batch, eta)
            if test_data:
                print("Epoch {} : {} / {}".format(j, self.accuracy(test_data), len(test_data)))
            else:
                print("Epoch {} finalizada".format(j))

    def update_mini_batch(self, mini_batch, eta):
        """Update the network's weights and biases using a mini-batch of training data.

        Args:
            mini_batch: list of tuples (x, y) where x is a vector of inputs
                        and y is a vector of target outputs
            eta: learning rate

        Returns:
            The final state of the network after training.
        """
        nabla_b = [np.zeros(b.shape) for b in self.biases]
        nabla_w = [np.zeros(w.shape) for w in self.weights]

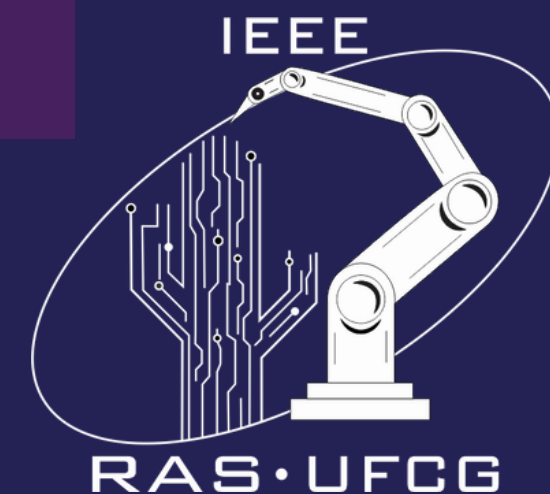
        for x, y in mini_batch:
            delta_nabla_b, delta_nabla_w = self.backprop(x, y, nabla_b, nabla_w)

            nabla_b = [nb + db for nb, db in zip(nabla_b, delta_nabla_b)]
            nabla_w = [nw + dw for nw, dw in zip(nabla_w, delta_nabla_w)]

        self.biases = [b + eta * nb for b, nb in zip(self.biases, nabla_b)]
        self.weights = [w + eta * nw for w, nw in zip(self.weights, nabla_w)]

```

TrainIEEE RAS Python



Quem Somos



Alana Nóbrega

Engenharia Elétrica (UFCG)

Coordenadora da equipe do
Braço Robótico -RAS
Web Developer FullStack



Giovana Oliveira

Ciência da Computação (UFCG)

Monitora de Jogos Digitais (2019)
e Paradigmas de Linguagens de
Programação (2020)



Lara Sobral

Engenharia Elétrica (UFCG)

Tesoureira do Ramo Estudantil
IEEE (UFCG)
Membro do Academy Translate

A Ferramenta



colab

- Plataforma gratuita
- Voltada à ciência da dados
- Sem necessidade de instalação na máquina

<https://colab.research.google.com>

A Proposta do Curso

Aprender os conteúdos enquanto os aplica num projeto
(estilo bootcamp)

O projeto: jogo adivinhe o número



Assuntos Abordados

- 1 - Variáveis, entrada e saída
- 2 - Estruturas condicionais (if, elif e else)
- 3 - Estruturas de repetição (for e while)
- 4 - Listas
- 5 - Funções

Expressões Matemáticas

- Adição (+)
- Subtração (-)
- Multiplicação (*)
- Potenciação (**)
- Divisão com decimal (/)
- Divisão inteira (//)
- Resto (%)
- OBS: utilização da regra matemática , multiplicação e divisão prioritários
- OBS2: Utilização de parênteses

```
def __init__(self, sizes):
    self.num_layers = len(sizes)
    self.sizes = sizes
    self.biases = [np.random.randn(y, 1) for y in sizes[1:]]
    self.weights = [np.random.randn(y, x) for x, y in zip(sizes[:-1], sizes[1:])]

def feedforward(self, a):
    for b, w in zip(self.biases, self.weights):
        a = sigmoid(np.dot(w, a)+b)
    return a

def SGD(self, training_data, epochs, mini_batch_size,
        training_data = training_data)
    n = len(training_data)

    for j in range(epochs):
        if test_data:
            print("Epoch {} : {} / {}".format(j, self.cost, self.accuracy))

        for k in range(n//mini_batch_size+1):
            mini_batch = training_data[k:k+mini_batch_size]
            for mini_batches:
                self.update_mini_batch(mini_batch, eta)
            if test_data:
                print("Epoch {} : {} / {}".format(j, self.cost, self.accuracy))
            else:
                print("Epoch {} finalizada".format(j))

def update_mini_batch(self, mini_batch, eta):
    nabla_b = [np.zeros(b.shape) for b in self.biases]
    nabla_w = [np.zeros(w.shape) for w in self.weights]

    for x, y in mini_batch:
        delta_nabla_b, delta_nabla_w = self.backprop(x, y)
        nabla_b = [nb+dnb for nb, dnb in zip(nabla_b, delta_nabla_b)]
        nabla_w = [nw+dnw for nw, dnw in zip(nabla_w, delta_nabla_w)]
```

Tipos de Dados

- int (número inteiro)
- float (número flutuante)
- booleana (true or false)
- Resposta do input com tipagem definida String, deve-se fazer conversão (cast).

tipo(input("Digite Texto"))

```
def __init__(self, sizes):
    self.num_layers = len(sizes)
    self.sizes = sizes
    self.biases = [np.random.randn(y, 1) for y in sizes[1:]]
    self.weights = [np.random.randn(y, x) for x, y in zip(sizes[:-1], sizes[1:])]

def feedforward(self, a):
    for b, w in zip(self.biases, self.weights):
        a = sigmoid(np.dot(w, a)+b)
    return a

def SGD(self, training_data, epochs, mini_batch_size,
        training_data_index, test_data_index):
    n = len(training_data)

    for j in range(epochs):
        if test_data_index:
            test_data = training_data[test_data_index:]

        for k in range(n//mini_batch_size):
            mini_batch = training_data[training_data_index+k*mini_batch_size:
                                       training_data_index+(k+1)*mini_batch_size]
            for mini_batch_size in range(mini_batch_size):
                self.update_mini_batch(mini_batch, eta)
            if test_data_index:
                print("Epoch {} : {} / {}".format(j, self.cost_function(training_data),
                                                    self.cost_function(test_data)))
            else:
                print("Epoch {} finalizada".format(j))

def update_mini_batch(self, mini_batch, eta):
    nabla_b = [np.zeros(b.shape) for b in self.biases]
    nabla_w = [np.zeros(w.shape) for w in self.weights]

    for x, y in mini_batch:
        delta_nabla_b, delta_nabla_w = self.backprop(x, y)
        nabla_b = [nb+dnb for nb, dnb in zip(nabla_b, delta_nabla_b)]
        nabla_w = [nw+dnw for nw, dnw in zip(nabla_w, delta_nabla_w)]
```


Comparações

- $(n > m)$
- $(n < m)$
- $(n == n)$
- $(n != m)$
- OBS: Respostas Booleanas

Expressões Booleanas

- or (ou) true
- and (e) false
- not (não)

```
def __init__(self, sizes):
    self.num_layers = len(sizes)
    self.sizes = sizes
    self.biases = [np.random.randn(y, 1) for y in sizes[1:]]
    self.weights = [np.random.randn(y, x) for x, y in zip(sizes[:-1], sizes[1:])]

def feedforward(self, a):
    for b, w in zip(self.biases, self.weights):
        a = sigmoid(np.dot(w, a)+b)
    return a

def SGD(self, training_data, epochs, mini_batch_size,
        training_data_index, test_data_index):
    n = len(training_data)

    for j in range(epochs):
        if test_data_index:
            test_data = training_data[test_data_index:]

        for k in range(n//mini_batch_size+1):
            mini_batch = training_data[training_data_index+k*mini_batch_size:
                                       training_data_index+(k+1)*mini_batch_size]
            for mini_batch_size in range(mini_batch_size):
                self.update_mini_batch(mini_batch, eta)
            if test_data_index:
                print("Epoch {} : {} / {}".format(j,self.feedforward(test_data[0]),
                                                    self.feedforward(test_data[-1])))
            else:
                print("Epoch {} finalizada".format(j))

def update_mini_batch(self, mini_batch, eta):
    nabl_a_b = [np.zeros(b.shape) for b in self.biases]
    nabl_a_w = [np.zeros(w.shape) for w in self.weights]

    for x, y in mini_batch:
        delta_nabl_a_b, delta_nabl_a_w = self.backprop(x, y)
        nabl_a_b = [nb+dnb for nb, dnb in zip(nabl_a_b, delta_nabl_a_b)]
        nabl_a_w = [nw+dnw for nw, dnw in zip(nabl_a_w, delta_nabl_a_w)]
```


1 - Variáveis, Entrada e Saída

Como conversar com o computador através da linguagem Python 3?



Entrada

`input()`

(diferenças entre input e raw_input)

Saída

`print()`

Criação de Variáveis

`num = 1`

`nome = "fulano"`

`entrada = input("Digite algo: ")`

Exemplo 1

Faça um programa Python que recebe o primeiro nome, o último nome e a idade de alguém e imprime: Bem vindo XXX XXX de XX anos!

```
# Recebe o nome, sobrenome e idade do jogador
primeiro_nome = input("Digite seu primeiro nome: ")
ultimo_nome = input("Digite seu último nome: ")
idade = input("Digite sua idade: ")

# Imprime as boas vindas
print("Bem vindo(a), " + primeiro_nome + " " + ultimo_nome + " de " +
      idade + " anos!")
```

Fazendo um Jogo em Python: Fase 1

Na primeira fase devemos perguntar o nome do jogador, imprimir na tela uma mensagem de boas vindas contendo o seu nome e receber um inteiro do personagem.

Conceitos abordados:

- Variáveis;
- Entrada e saída;
- Concatenação de strings;

Fazendo um Jogo em Python: Fase 1

```
nome = input("Digite seu nome: ")  
print("*"*100)  
print("Bem vindo(a), " + nome + "!\n" + "Será se você consegue adivinhar  
o número secreto?")  
print(("*"*100) + "\n")
```

2 - Estruturas Condicionais (if, elif, else)

Condição Inicial

if

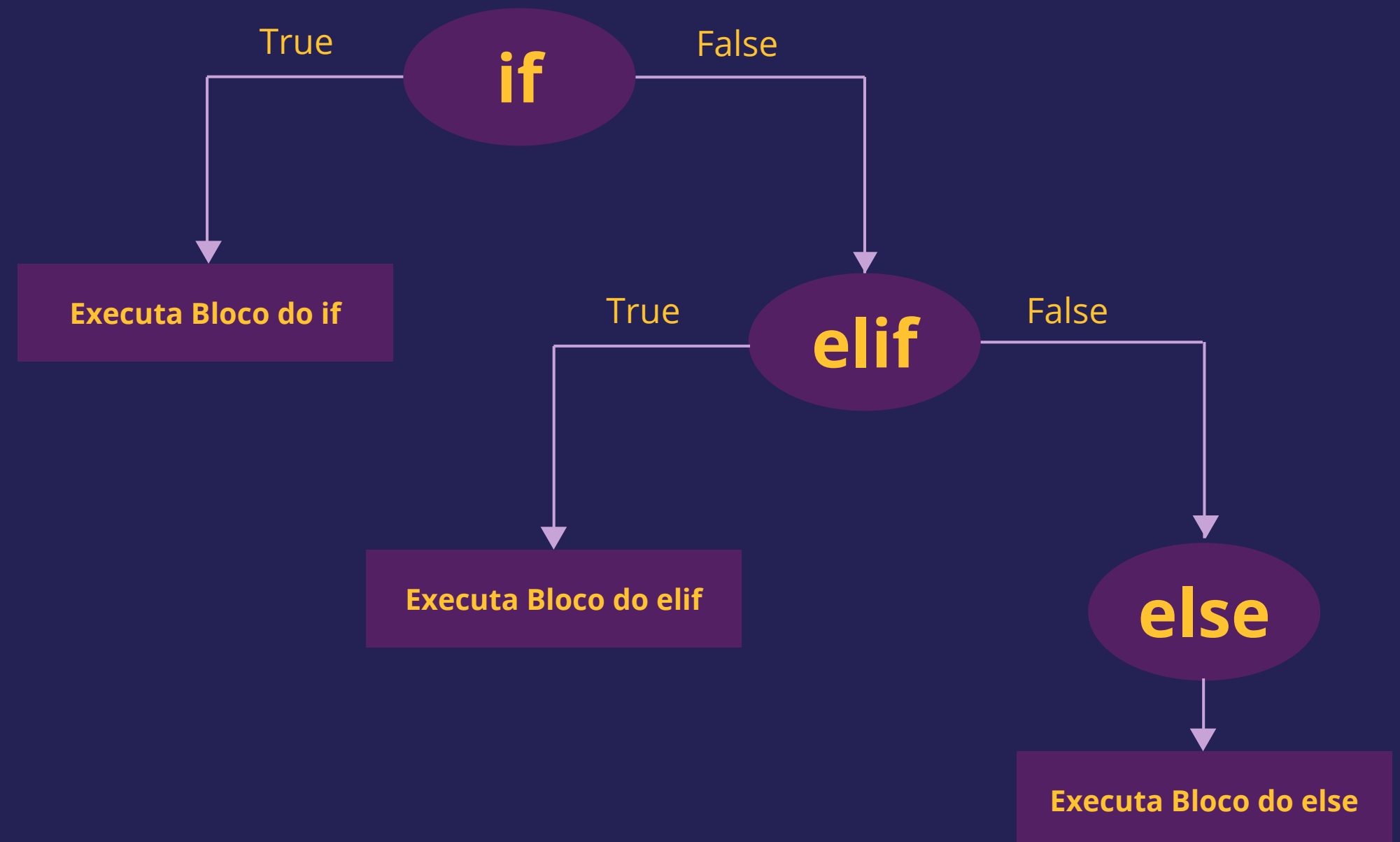
Condição Intermediária

elif

(quando tem mais de duas condições testadas)

Caso Contrário

else



2 - Estruturas Condicionais (if, elif, else)

Sintax

```
if condição:  
    bloco de código do if  
elif condição:  
    bloco de código do elif  
else:  
    bloco de código do else
```

Identação é essencial em Python

Exemplo 2

Faça um programa Python que recebe um inteiro que imprime na tela se o número é ímpar ou par:

```
# Recebe um inteiro qualquer do usuário
num = int(input("Digite um número inteiro: "))

# Checa se o número é par ou ímpar
if (num % 2) == 0:
    print("é par!")
else:
    print("é ímpar!")
```


Importando Bibliotecas

```
from random import randint
```

```
num = randint(0,9)
```

Fazendo um Jogo em Python: Fase 2

Nosso jogo deve ser capaz de dizer se o inteiro recebido do jogador é **menor** que o esperado, **maior**, ou se ele **acertou o número secreto** gerado randomicamente.

Caso o número recebido não esteja entre 0 e 9 "**Entrada inválida!**" deve ser impresso na tela.

Conceitos aplicados:

- Estruturas condicionais (if, elif e else);
- Conversão de tipo;
- Geração randômica de números;
- Validação.

Fazendo um jogo em Python: Fase 2

```
from random import randint
num_sec = randint(0,9)
num = (int)(input("Digite um numero de 0 a 9: \n"))
if num < 0 or num > 9:
    print("Entrada inválida!")
elif num < num_sec:
    print("Número muito baixo!")
elif num > num_sec:
    print("Número muito alto!")
else:
    print("Você acertou!")
```

3 - Estruturas de repetição (for e while)

**Sei quantas vezes
preciso repetir?**

for n in range(x, y):

(x condição inicial condição final)

**Não sei quantas
vezes preciso repetir?**

while ():



3 - Estruturas de repetição (for e while)

Sintax (for)

```
for variavel in lista:  
    comandos
```

```
nomes = ['Pedro', 'João', 'Leticia']  
for n in nomes:  
    comandos
```

```
for i in range(0, 10):  
    comandos
```

Exemplo 3

(for)

Faça um programa Python que recebe um inteiro e imprime na tela o fatorial dele:

```
num = int(input("Digite um número: "))
fatorial = num

for i in range(num - 1, 1, -1):
    fatorial *= i
print(fatorial)
```

3 - Estruturas de repetição (for e while)

Sintax (while)

```
while condição:  
    comandos
```


Exemplo 3

(while)

Faça um programa Python que recebe um inteiro e imprime na tela o fatorial dele:

```
num = int(input("Digite um número: "))
fatorial = num

while num > 1:
    num -= 1
    fatorial *= num

print(fatorial)
```

Fazendo um jogo em Python: Fase 3

Nosso jogo deve ser capaz de continuar pedindo o número do jogador até que ele acerte o número secreto.

Conceito aplicado:

- Estruturas de repetição

Fazendo um jogo em Python: Fase 3

```
achou = False
while not achou:
    num = (int)(input("Digite um numero de 0 a 9: \n"))
    if num < 0 or num > 9:
        print("Entrada inválida! Tente novamente.\n")
    elif num < num_sec:
        print("Número muito baixo, tente um maior!")
    elif num > num_sec:
        print("Número muito alto, tente um menor!")
    else:
        achou = True
        print("Você acertou!")
```

4 - Listas



Lista em Branco

```
lista = [ ]
```

(lista a ser preenchida)

Lista Preenchida

```
lista = ['item1', 'item2', 'item3']
```

(lista preenchida)

Iterando sobre listas

```
lista = ["ana", "maria", "julia"]  
for i in range(len(lista)):  
    print(lista[i])
```

Exemplo 4

Faça um programa Python que indica quantos alunos existem numa turma, recebe o nome de cada um deles, os coloca numa lista e depois imprime a lista.

```
n = int(input("Digite a quantidade de alunos: "))
alunos = []
for i in range(n):
    aluno = input()
    alunos.append(aluno)

print(str(alunos))
```

Fazendo um jogo em Python: Fase 4

Quais números nosso jogador já tentou? Nosso jogo irá armazenar todas as tentativas do usuário e disponibilizar essa lista em tela.

Conceito aplicado:

- Listas.

Fazendo um jogo em Python: Fase 4

```
achou = False
tentativas = []
while not achou:
    num = (int)(input("Digite um numero de 0 a 9: \n"))
    if num < 0 or num > 9:
        print("Entrada inválida! Tente novamente.\n")
    elif num < num_sec:
        print("Número muito baixo, tente um maior!")
        tentativas.append(num)
        print("Números que você já testou: " + str(tentativas) + "\n")
    elif num > num_sec:
        print("Número muito alto, tente um menor!")
        tentativas.append(num)
        print("Números que você já testou: " + str(tentativas) + "\n")
    else:
        achou = True
        print("Você acertou!")
```

5 - Funções



Definindo Função

```
def funcao ( ):
    return
```

Exemplo 5

Faça um programa Python que calcula o IMC, recebe os dados de altura e peso e retorna o valor calculado.

```
altura = float(input("Digite a altura: "))
peso = float(input("Digite o peso: "))

def calcular_imc (altura, peso):
    x = peso/(altura**2)
    return x

print(calcular_imc(altura, peso))
```

Fazendo um jogo em Python: Fase 5

Nosso jogo precisa ser refatorado! Vamos criar duas funções que iram dividir nosso código em blocos. A primeira função ira executar o bloco de boas-vindas. E a segunda função ira executar o nosso loop de tentativa e erro.

Conceito aplicado:

- Função.

Fazendo um jogo em Python: Fase 5

```
def boas_vindas(nome):  
    print("*"*100)  
    print("Bem vindo " + nome + "!\n" + "Será se você consegue adivinhar  
o número secreto?")  
    print(("*"*100) + "\n")
```

Fazendo um jogo em Python: Fase 5

```
def analise(num):  
    achou = False  
    tentativas = []  
    while (not achou):  
        num = (int)(input("Digite um numero de 0 a 9: \n"))  
        if num < 0 or num > 9:  
            print("Entrada inválida! Tente novamente.\n")  
        elif num < num_sec:  
            print("Número muito baixo, tente um maior!")  
            tentativas.append(num)  
            print("Números que você já testou: " + str(tentativas) + "\n")  
        elif num > num_sec:  
            print("Número muito alto, tente um menor!")  
            tentativas.append(num)  
            print("Números que você já testou: " + str(tentativas) + "\n")  
        else:  
            achou = True  
            print("Você acertou!")
```

Fazendo um jogo em Python: Fase 5

```
from random import randint

# Recebe nome e imprime boas vindas
nome = input("Digite seu nome: ")
boas_vindas(nome)

# Analisa se o número recebido está correto
num_sec = randint(0,9)
analise(num_sec)
```

Em Python primeiro define a função depois chama.

Embarcando na Jornada da Programação

Algumas dicas importantes:

- Antes de programar, tenha em mente seu pseudocódigo, para lhe orientar durante a programação
- Sempre confira a indentação do seu código em Python
- Em Python, na declaração de um estrutura lembre de finalizar com ":"
- Comentários (#) auxiliam na compreensão do código, tanto para quem escreve quanto para quem for lê-lo.
- Tenha um Git Hub organizado e um portfólio dos seus trabalhos realizados

Por Onde Eu Posso Aprender Python?

- Python For Everybody do Charles Severance, disponível em pt.coursera.org/
- Curso de Python 3 do Gustavo Guanabara, disponível em youtube.com/c/CursoemVideo/
- Livro Python For Everibody do Charles Severance, disponível em py4e.com/book.php
- Livro Pense em Python do Allen B. Downey
- Documentação oficial da linguagem Python 3, disponível em: <http://docs.python.org/pt-br/3/>

```

def __init__(self, sizes):
    self.num_layers = len(sizes)
    self.sizes = sizes
    self.biases = [np.random.randn(y, 1) for y in sizes[1:]]
    self.weights = [np.random.randn(y, x) for x, y in zip(sizes[:-1], sizes[1:])]

```

```

def feedforward(self, a):
    for b, w in zip(self.biases, self.weights):
        a = sigmoid(np.dot(w, a)+b)
    return a

```

```

def SGD(self, training_data, epochs, mini_batch_size,
        training_data_index, test_data):
    n = len(training_data)

```

```

    if test_data:
        test_results = []

    for j in range(epochs):
        training_data_index = random.shuffle(training_data)
        mini_batches = [training_data[alk:k+mini_batch_size]
                        for alk in range(0, n, mini_batch_size)]
        for mini_batch in mini_batches:
            self.update_mini_batch(mini_batch, eta)
        if test_data:
            print("Epoch {}: {} / {}".format(j, self.evaluate(test_data), n))
        else:
            print("Epoch {} finalizada".format(j))

```

```

def update_mini_batch(self, mini_batch, eta):
    nabla_b = [np.zeros(b.shape) for b in self.biases]
    nabla_w = [np.zeros(w.shape) for w in self.weights]

    for x, y in mini_batch:
        delta_nabla_b, delta_nabla_w = self.backprop(x, y, nabla_b, nabla_w)
        nabla_b = [nb+dnb for nb, dnb in zip(nabla_b, delta_nabla_b)]
        nabla_w = [nw+dnw for nw, dnw in zip(nabla_w, delta_nabla_w)]

```

Obrigada pela participação!



bit.ly/34evVW1 (repositório do curso)



@sbc.rasufcg



@sbcrasufcg



/company/ieee-ras-ufcg/



edu.ieee.org/br-ufcgras/ras/