TEMAS EM BIOINFORMÁTICA

De 10 à 14 de agosto de 2020

Por que aprender a programar? Por que Python?

Prof. Dr. Gilderlanio Santana de Araújo
Laboratório de Genética Humana e Médica - UFPA
gilderlanio@gmail.com

1

Quem sou eu?

Prof. Dr. Gilderlanio Santana de Araújo
Visitante do Laboratório de Genética Médica (UFPA) - Pará
Doutor em Bioinformática (UFMG) – Minas Gerais
Mestre em Ciências da Computação (UFPE) - Pernambuco
Bacharel em Sistemas de Informação (UNIFIP) - Paraíba

2

Elementos da programação de computadores

3

Programa de computador

Conjunto de instruções em linguagens de programação

4

Algoritmos

"Um procedimento passo a passo para a solução de problemas."

"Uma sequência detalhada de ações a serem executadas para realizar alguma tarefa."

5

Algoritmos

Algoritmo: Somar três números
1. Receber os três números
2. Somar os três números
3. Mostrar o resultado obtido

6

Algoritmos

Algoritmo: Trocar uma lâmpada
1. Comprar uma lâmpada nova
2. Retirar a lâmpada queimada
3. Colocar a lâmpada nova
4. Testar o interruptor

7

Como construir algoritmos?

- Compreender o problema a ser resolvido!
- Destaque os pontos mais importantes e os objetos que compõe o problema.
- Defina os dados de entrada.
- Defina o processamento.
- E se o problema for complexo?
  - Divida pra conquistar!
- Defina seus dados de saída.

8

Algoritmos

Algoritmo: Trocar uma lâmpada
1. Comprar uma lâmpada nova
2. Retirar a lâmpada queimada
3. Colocar a lâmpada nova
4. Testar o interruptor

9

Para representar um algoritmo,

DESCREVA!

10

Construção de algoritmos

- Por linguagem natural
- Por fluxogramas
- Por pseudocódigos

11

Construção de algoritmos

- Por linguagem natural
- Por fluxogramas
- Por pseudocódigos

Algoritmo 1 Exemplo de Pseudocódigo.

```
leia (x, y)  {Esta linha é um comentário}
se  x > y  então
    escreva  ("x é maior")
senão
    se y > x  então
        escreva  ("y é maior")
    senão
        escreva  ("x e y são iguais")
    fim-se
fim-se
```

12

Pseudocódigos

- Código intermediário em uma linguagem independente das peculiaridades dos compiladores.
- Adição de detalhes de linguagens de programação
- Mais fácil de traduzir para qualquer linguagem de programação.

13

Pseudocódigos

Algoritmo 1 Exemplo de Pseudocódigo.

```
leia (x, y) {Esta linha é um comentário}
se x > y então
    escreva ("x é maior")
senão
    se y > x então
        escreva ("y é maior")
    senão
        escreva ("x e y são iguais")
    fim-se
fim-se
```

14

Por que aprender a programar?

15

- Se destacar profissionalmente;
- Resolução de problemas complexos;
- Organização pessoal;
  - Saber programar ajuda a ganhar tempo;
- Autonomia para automatizar suas tarefas;
- Colaboração em projetos;
- Dados em grandes volumes.

16

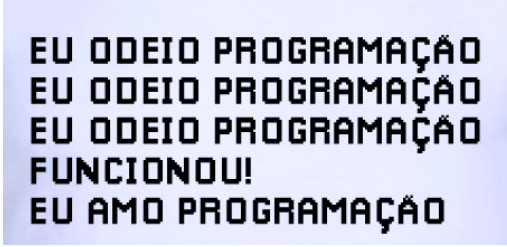
Como aprender a programar?

17

Não há receita! Entretanto,

- Entenda e pratique lógica de programação
- Escolha uma linguagem!
- Desafie-se e crie um projeto!
- Seja resiliente!
  - "Não se aperrei!"
  - "Não te bate!"

18

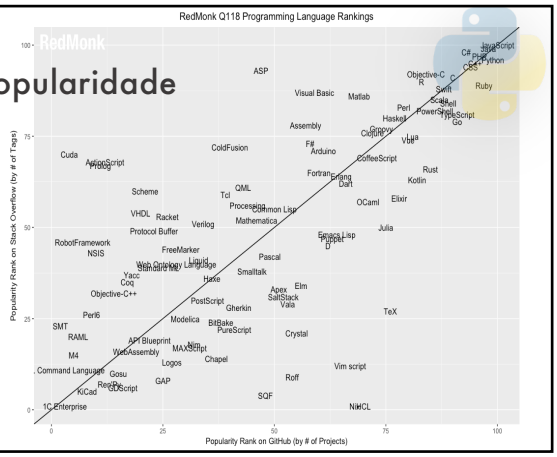


19

Por que Python?

20

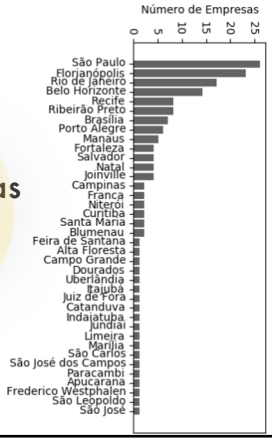
Popularidade



21

Cidades pythônicas

(https://python.org.br/empresas/)

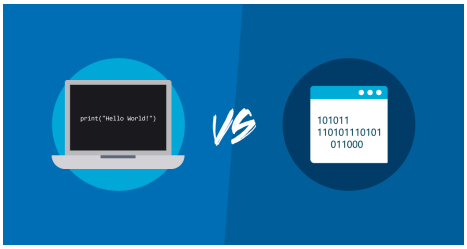


22

Python

- Criada em 1990 por Guido Van Rossum
- Versões 2.x e 3.x
- Linguagem de alto nível
- Interpretada
- Pode ser executada via script ou no modo interativo

23



24

Linguagem de propósito geral

25

Paradigmas de linguagem

- Python foi construída sob características de diversos paradigmas de linguagem

26

Características

- Sintaxe clara e concisa;
- Organização do código por identação e ":";
- Vários tipos de dados (inteiros, decimais, cadeias de caracteres);
- Estruturas de dados robustas (conjuntos, dicionários);
- Modularidade
- Interpretada

27

Variáveis e tipos de dados

- Entidades definidas por um nome.
- Referem-se a um tipo de dado
- Podem mudar seu conteúdo em tempo de execução

28

Variáveis e tipos de dados

- Inteiros
  - Ex.: -1, 0, 1, 2, 3, 4, 5
- Decimais
  - Ex.: 0.0, 1.5, 10.5, 3.2
- Booleanos
  - Ex. Verdadeiro ou falso
- Sequências de caracteres
  - Ex. "Aprenda Python!"

29

Operações aritméticas

- adição e concatenação (+)
- subtração (-)
- multiplicação (*)
- exponenciação (**)
- divisão (/)
- divisão inteira(//)
- módulo (%)

30

Operações lógicas

- < (menor que);
- > (maior que);
- == (igual);
- <= (menor ou igual);
- >= (maior ou igual);
- ! = (diferente).

- and
- or
- not
- In
- not in

31

Estruturas de dados

- Listas (list())
  - x = [1, 2, 3, 4, 5]
  - genes = ['FTO', 'APOE', 'BRCA1']
- Tuplas (tuple())
  - x = (1, 'Gil', 'Professor')
- Conjuntos (set())
  - set([1, 2, 3, 4, 5])
- Dicionários (dict())
  - genes = dict()
  - genes = {'FTO' : {'inicio' : 12345, fim : '54321'}}
  - print(genes['FTO']) ??
  - print(genes['FTO']['inicio']) ??

32

Sumário dos tipos de dados

Tipo de dado	Complexidade	Em ordem	Mutável	Indexado	Heterogêneo
int	primitivo	–	sim	–	–
float	primitivo	–	sim	–	–
complex	primitivo	–	sim	–	–
Boolean	primitivo	–	sim	–	–
string	contêiner	sim	não	sim	não
list	contêiner	sim	sim	sim	sim
tuple	contêiner	sim	não	sim	sim
set	contêiner	não	não	não	sim
dictionary	contêiner	não	sim	não	sim

33

Estruturas condicionais

- As estruturas condicionais auxiliam a tomar decisões em tempo de execução.

```
if test1:
    statements1
elif test2:
    statements2
```

34

Estruturas de repetição

- Condicionais
  - while

```
while test:
    statements
else:
    statements
```

- Interativas
  - for

```
for target in object:
    statements
else:
    statements
```

35

Funções

- Built-in
  - len()
  - range()
- Módulos
  - Math
  - Biopython
- Crie suas próprias funções
  - Comando *def*

```
def printer(messge):
    print('Hello ' + message)
```

36

BioPython

- Projeto de código aberto
- Diversos módulos para biologia computacional
- Open Bioinformatics Foundation (OBF)
- Website: https://biopython.org/

37

Áreas de abrangência

- Manipulação de sequências
  - Alinhamento par-a-par
  - Alinhamento multiplo
- Genética de Populações
- Bioinformática Estrutural
- Algoritmos de agrupamento
- Acesso a diversas bases de dados
- Acesso a ferramentas bioinformáticas

38

Interface com softwares e serviços web

- BLAST: standalone e web
- Clustalw
- FASTA
- GenBank
- PubMed and Medline
- ExPASy files, like Enzyme and Prosite
- SCOP, including 'dom' and 'lin' files
- UniGene
- SwissProt

39

Desenvolvendo scripts em Python

Ambientes de desenvolvimento integrado.

40

Desenvolvendo scripts em Python

Ambientes de desenvolvimento integrado.

41

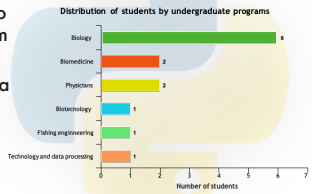
Sumário

- Robustez para sistemas complexos
- Vários tipos de objetos
- Várias funções implementadas
- Fácil de aprender
- Fácil de usar
- Fácil de codificar

42

Experiências

- Disciplina: Programação para Bioinformática com Python
- PPG Genética e Biologia Molecular - UFPA
- 26 alunos (medicina, biologia, biomedicina, biotecnologia, ...)

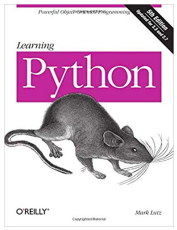


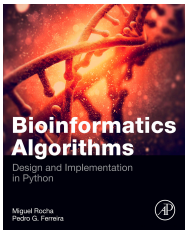
43

Experiências

44

Livros de referência





45

Recursos online.

- Python. https://www.python.org/
- Python Brasil. https://python.org.br/
- BioPython. https://biopython.org/

46

Redes sociais

Twitter e Instagram

@lghm_ufpa
www.lghm.com.br

Twitter : @gilderlanio_
gilderlanio@gmail.com

47

while true:
print("Obrigado!")

https://www.lghm.com.br/

48