



AI Talks

Autograde com IA e IA no Ensino de Programação Básica

Alex Pereira – COMP 04

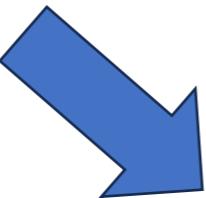
Resolvendo Problemas com o ChatGPT

Bob Jane Ivy John Laura Greg

| | | | | | |
|---|----|----|---|----|----|
| 5 | 10 | 30 | 0 | 35 | 40 |
|---|----|----|---|----|----|

keys_in_range(1, 9, 50) =

["Jane", "Ivy", "Laura", "Greg"]



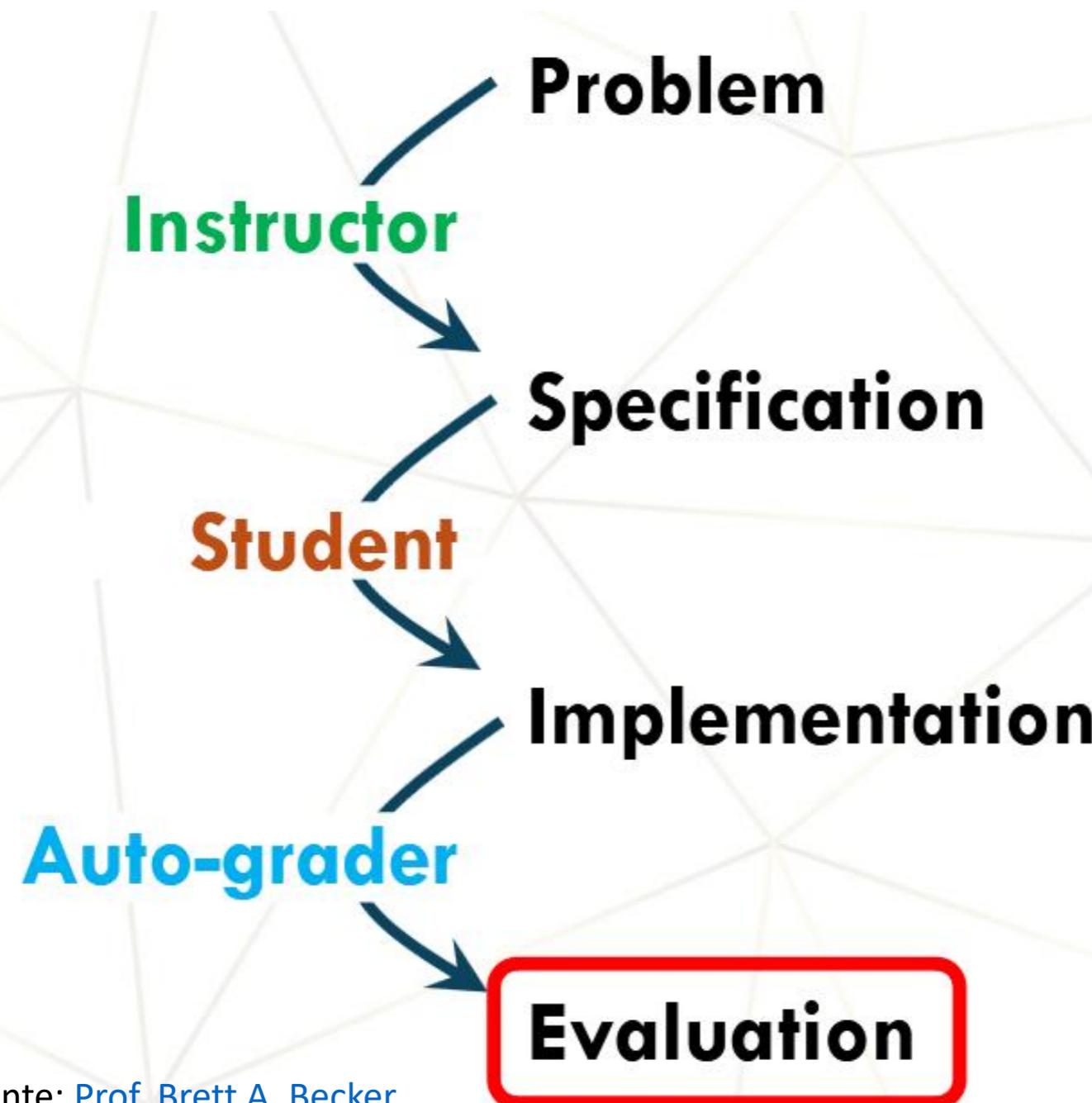
python

Copy code

```
def keys_in_range(data, lower, upper):
    return [key for key, value in data.items() if lower <= value <= upper]

data = {"Bob": 5, "Jane": 10, "Ivy": 30, "John": 0, "Laura": 35, "Greg": 40}
result = keys_in_range(data, 9, 50)
print(result) # Output: ['Jane', 'Ivy', 'Laura', 'Greg']
```

Ensino de Programação Básica

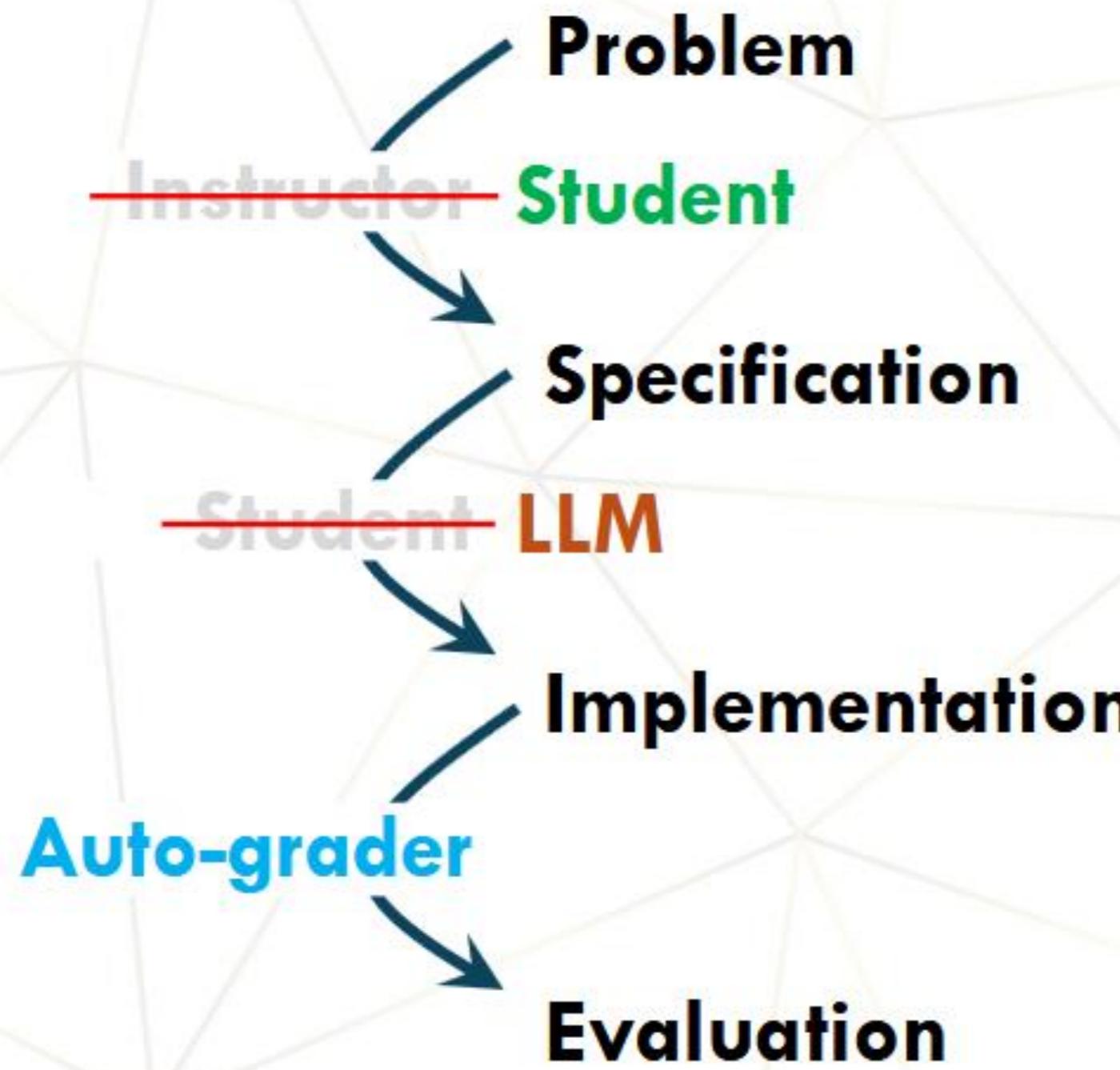


"This is good" → "This is ****"
"good"

A sentence can be "censored" by having all banned words removed. Define a function called `censor_sentence()` which takes two inputs: a sentence (this will be a string, with no punctuation, where words are separated by a single space character) and a list of banned words. The function should return a new string where all of the characters in any banned word are replaced with "*".

```
def censor_sentence(sentence, banned_words):
    sentence = sentence.split()
    for word in sentence:
        if word in banned_words:
            sentence[sentence.index(word)] = "*" * len(word)
    return " ".join(sentence)
```

| | |
|---|--|
| ✓ | s = "apple banana cherry dragonfruit" cs = censor_sentence(s, ["dragon", "fruit", "cherry"]) print(cs) |
| ✓ | s = "a aa aaa aaaa aaaaa" cs = censor_sentence(s, ["a", "aaa"]) print(cs) |
| ✓ | s = "a a a a a a a a" cs = censor_sentence(s, ["a"]) print(cs) |



"This is good" → "This is ****"
"good"

A sentence can be "censored" by having all banned words removed. Define a function called **censor_sentence()** which takes two inputs: a sentence (this will be a string, with no punctuation, where words are separated by a single space character) and a list of banned words. The function should return a new string where all of the characters in any banned word are replaced with "*".

```
def censor_sentence(sentence, banned_words):
    sentence = sentence.split()
    for word in sentence:
        if word in banned_words:
            sentence[sentence.index(word)] = "*" * len(word)
    return " ".join(sentence)
```

| | |
|---|--|
| ✓ | s = "apple banana cherry dragonfruit" cs = censor_sentence(s, ["dragon", "fruit", "cherry"]) print(cs) |
| ✓ | s = "a an ana anna annaa" cs = censor_sentence(s, ["a", "aa"]) print(cs) |
| ✓ | s = "***** a" cs = censor_sentence(s, ["a"]) print(cs) |

Contexto Pós IA Generativa

- Não faz mais sentido avaliar Código de um exercício respondido por um Aluno
 - **Devemos alinhar a avaliação à prática**
 - ✓ Queremos que as pessoas usem IA? => Avaliemos elas no uso da IA
 - **Avaliação incentiva/promove a prática de atividades (foco)**
 - ✓ Essas atividades devem ser
 - Compatíveis/equivalentes com a prática no trabalho do dia a dia
 - Cognitivamente atrativas/desafiadoras
- O que faz sentido agora é avaliar os prompts
 - **Avaliar se o aluno produziu um prompt**
 - ✓ que um LLM transforma em código e resolve o problema

Aplicação do Prof. Brett Becker

A screenshot of a web browser window. The address bar at the top shows the URL "promptly-sigcse.web.app". The main content area displays the text "Welcome to Promptly!". Below this, there is a blue link that says "CLICK HERE TO LOG OUT!". At the bottom of the page, there is a paragraph of text in Portuguese. The browser interface includes a toolbar with various icons for file operations like cut, copy, paste, and save, as well as a search bar and a refresh button. The overall layout is clean and professional.

Problema (Professor)

```
counter([0, 2, 3, 4, 5, 6, 0]) => 2
```

```
counter([10, 20, 30]) => 0
```

```
counter([0, 0, 0, 0, 999]) => 4
```

Especificação (Aluno)

Escreva-me uma função Python com o nome conte a quantidade de números inteiros igual a zero presentes numa lista passada como entrada da função counter.

Escreva aqui a sua pergunta para o ChatGPT

Word Count: 25

Implementação (LLM)

Resposta do ChatGPT:

```
def counter(lista):
    count = 0
    for num in lista:
        if num == 0:
            count += 1
    return count
```

Autograder

✓ Test 1

✓ Test 2

✓ Test 3

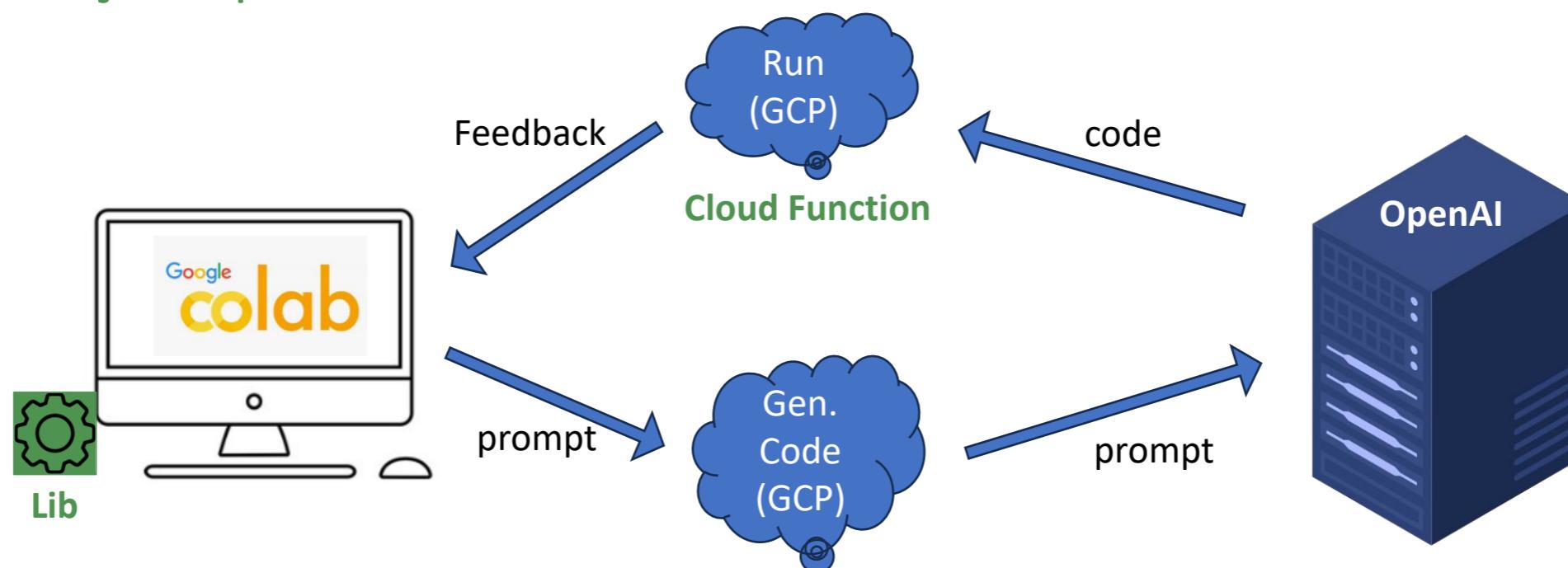
✓ Test 4

Resultado da execução do código

Passou! \(^o^)/

IDE: Ambiente de Desenvolvimento e aprendizado

- A avaliação deveria estar dentro da IDE
 - Ou seja, Google Colab (curva de aprendizado / sustentação)
 - ✓ Para Ensino de programação
 - Solução Open Source

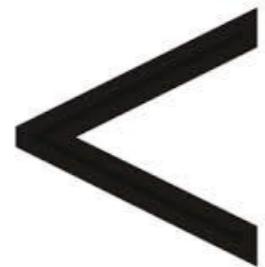
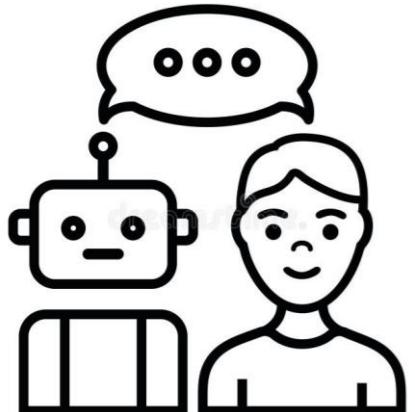


Componentes da Solução

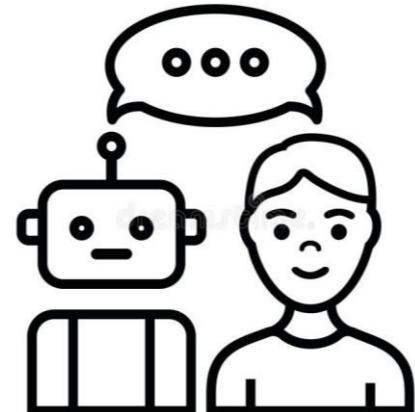
- Lib: [ipynb-autograde](#)
 - Coleta dados de context (user, exercício e prompt)
 - Envia para uma Cloud Function
 - Setup: exercícios, gabaritos, e curso
- GCF: [autogradeapi](#)
 - Converte prompt em Código usando um LLM
 - Executa o Código (com checagens de segurança)
 - Setup: Chave de API, Alunos Autorizados, autenticação pelo token do colab
 - IA ajudou na arquitetura e implementação
 - ✓ Da autenticação e validações de segurança
- Dashboard (Looker Studio) + Google Sheets

Conclusões preliminaries (viés de seleção – educação continuada)

Aprendizado
com Grade de
Código
(gerados por um LLM)



Aprendizado
com Grade de
Prompt
(gerados por um LLM)



Aprendizado
Sem IA



Exercício de Reflexão corrigido por IA

- Os alunos começaram a usar a IA para elaborar o texto de reflexão
 - O mesmo que treinar corrida, mas treinar indo de carro.

Exercício de Reflexão (Facultativo)

Você aprendeu alguma coisa que pode ser reaproveitado em outros problemas?

Como você pode melhorar seu processo de elaboração de requisitos / prompts?

Você notou ambiguidade na sua especificação inicial?

```
resposta="""AQUI"""
```

```
validate(resposta, "A2-R1")
```

O futuro é de quem sabe pedir

Na Era da IA

Quem sabe pedir é quem tem o vocabulário necessário