Evaluating Curriculum Rigor

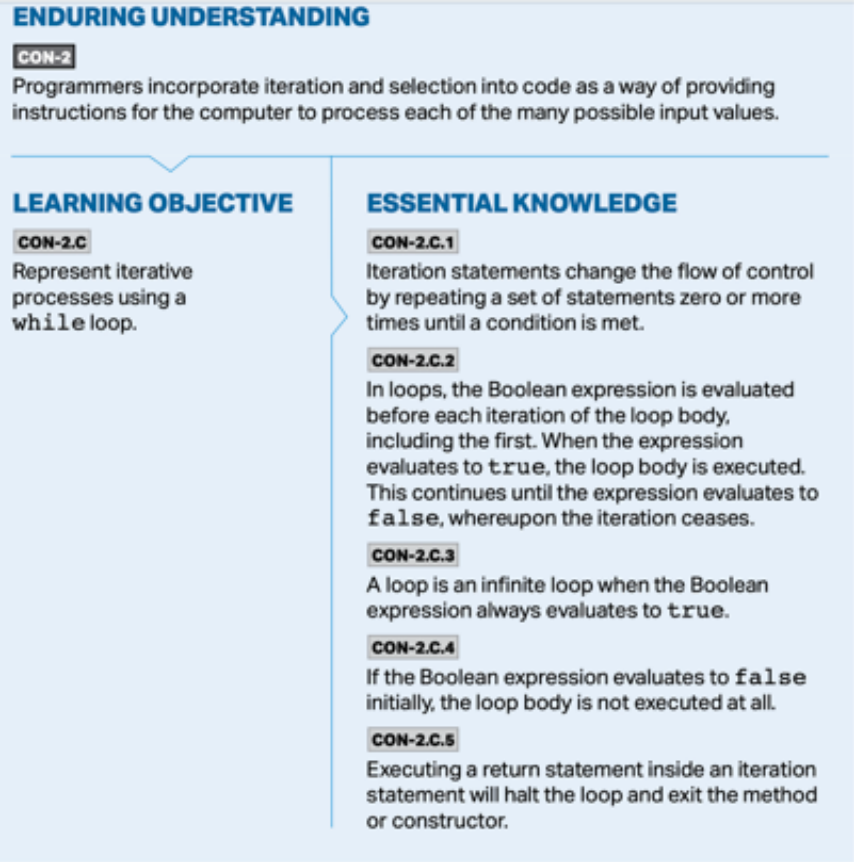
Background

In my experience with high school curriculum, I have found a wide variation in the rigor of course material. This project seeks to develop a tool for evaluating the rigor of a curriculum, by measuring its alignment to the College Board's respective AP Course. This project focuses on the College Board's AP Computer Science A course, which covers a first year Java and Object Orientied Design course.

For this course, the College Board defines a set of "Computational Thinking Practices" (skills) and content that will be assessed on a year-end summative assessment to determine student's mastery of the course.

There are 5 main Computational Thinking Practices identified by the College Board, which it then breaks down into subskills:

In addition, the College Board defines a set of "Essential Knowledge" (the content) to be assessed in the course, which it organizes under 5 "Big Ideas." For example, the content for a lesson on iteration is:



ENDURING UNDERSTANDING

CON-2

Programmers incorporate iteration and selection into code as a way of providing instructions for the computer to process each of the many possible input values.

LEARNING OBJECTIVE

CON-2.C

Represent iterative processes using a `while` loop.

ESSENTIAL KNOWLEDGE

CON-2.C.1

Iteration statements change the flow of control by repeating a set of statements zero or more times until a condition is met.

CON-2.C.2

In loops, the Boolean expression is evaluated before each iteration of the loop body, including the first. When the expression evaluates to `true`, the loop body is executed. This continues until the expression evaluates to `false`, whereupon the iteration ceases.

CON-2.C.3

A loop is an infinite loop when the Boolean expression always evaluates to `true`.

CON-2.C.4

If the Boolean expression evaluates to `false` initially, the loop body is not executed at all.

CON-2.C.5

Executing a return statement inside an iteration statement will halt the loop and exit the method or constructor.

continued on next page

Every question on the College Board's end-of-course summative exam is aligned to a particular computational thinking skill and essential knowledge. As a note, some school networks have found the College Board's standards to be very complete, and "backwards plan" their middle school and pre-AP high school courses to prepare students for the AP level work.

As a first step, this project will focus on the assessment questions used in a particular curriculum, and measure how well they align to the College Board's Computational Thinking Practice and Curriculum Framework. (As a note, AP classes in most subjects have an analagous set of thinking practices and framework standards, so one day, this work may be generalized to assess curriculums in other subject areas.)

The three questions to assess are:

1. How accurately does a simple classifier using either a TF-IDF or SentenceTransformer vectorizer identify the Computational Thinking Practice assessed in a question?
2. How accurately does ChatGPT, supplied with only the College Board Framework for Computational Thinking and a few sample questions, identify the Computational Thinking Practice assessed in a question?

3. Can a useful Dashboard be created to improve the quality of an assessment question?

Initial Conclusions:

1. The classifier using a TF-IDF vectorization accurately identified the Computational Thinking Practice being accessed in 70% of the questions, while the simple classifier using the Sentence Transformed identified the Computational Thinking Practice being assessed in 85% of the questions.
2. ChatGPT had an accuracy of 48% in identifying the Computational Thinking Practice assessed in a question.

Next Steps:

1. Develop better prompts to improve ChatGPT's classification accuracy.
2. Determine whether the classifier can also identify the "Essential Knowledge" assessed by the question, not just the computational skill.
3. Attempt to generalize the classifiers to classify non-assessment questions such as lecture material, lab questions, and homework problems.
4. Create a visualization that shows the distribution of thinking skills and content assessed over the course of the curriculum.

Classifying Questions Using Logistic Regression

As first step, this section will try to classify prompts as assessing one of these two AP Computational Thinking Practices (CTP):

1. **CTP 2.A**: Apply the meaning of specific operators. For example:
   Consider the following code segment.

```
int x = 7;
int y = 3;
if ((x < 10) && (y < 0))
  System.out.println(""Value is: "" + x * y);
else
  System.out.println(""Value is: "" + x / y)
```

What is printed as a result of executing the code segment?

2. **CTP 2.B**: Determine the results or output based on statement execution order in a

code segment without method calls (except for output). For example:

Consider the following code segment.

```
int[] arr = {7, 2, 5, 3, 0, 10};
for (int k = 0; k < arr.length - 1; k++)  {
  if (arr[k] > arr[k + 1])
    System.out.print(k + "" "" + arr[k] + "" "");
}
```

What will be printed as a result of executing the code segment?

Preprocessing the Data

In this first step, we will:

1. Load questions assessing each Computational Thinking Practice.
2. Filter the question set to only consider the most frequently assessed Computational Thinking Practices.

In [5]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from sklearn.feature_extraction.text import TfidfVectorizer
from nltk.stem import WordNetLemmatizer
from nltk.tokenize import RegexpTokenizer
```

In [6]:
```
df = pd.read_csv("Data/synthetic_all_questions.csv")
df_2014 = pd.read_csv("Data/CollegeBoard/SamplePrompts-PracticeExam2014.csv
df_2014 = df_2014[["text","Classification"]]
df_2014.columns=["Question","Classification"]
df = pd.concat([df, df_2014])
```

In [7]:
```
df["Classification"] = df["Classification"].str.strip(" ':")
df = df[df["Classification"].isin(["1.B", "2.A", "2.B", "2.C", "2.D", "4.A"
```

Testing the Logistic Regression Classifier on the Questions

Here, we build and evaluate a classifier that vectorizes questions using the TF-IDF

vectorizer, and classifies using a Logistic Regression Classifier.

In [11]:
```
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LogisticRegression

from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import ConfusionMatrixDisplay

import nltk
```

In [12]:
```
basic_token_pattern = r"(?u)\b\w\w+\b"

tokenizer = RegexpTokenizer(basic_token_pattern)
lemmatizer = WordNetLemmatizer()

def lemmatize_text(text):
    temp = tokenizer.tokenize(text.lower())
    return [lemmatizer.lemmatize(w) for w in temp]

stopwords = nltk.corpus.stopwords.words("english")
stopwords.append("ha")
stopwords.append("doe")
stopwords.append("wa")

pipeline = [("tfidf",TfidfVectorizer(strip_accents='ascii', tokenizer=lemma
            ("lr", LogisticRegression(solver="saga", max_iter=10000))]
pipe = Pipeline(steps=pipeline)

params = [{"tfidf__max_features":[5,50,100], "lr__penalty":['l1', 'l2'],"lr
gs = GridSearchCV(estimator=pipe, param_grid=params)
```

In [13]:
```
X_train, X_test, y_train, y_test = train_test_split(df["Question"], df["Cla
gs.fit(X_train, y_train)
```

Out[13]:
```
GridSearchCV(estimator=Pipeline(steps=[('tfidf',
                                        TfidfVectorizer(max_features
=15,
                                                        stop_words=
['i', 'me',

'my',

'myself',
```

```
        'we', 'our',
        'ours',
        'ourselves',
        'you',
        "you're",
        "you've",
        "you'll",
        "you'd",
        'your',
        'yours',
        'yourself',
        'yourselves',
        'he', 'him',
        'his',
        'himself',
        'she',
        "she's",
        'her',
        'hers',
        'herself',
        'it',
        "it's",
        'its',
        'itself', ...],
```

```
                                                                strip_accent
s='ascii',
                                                                token_patter
n=None,
                                                                tokenizer=<f
unction lemmatize_text at 0x1557d1120>)),
                                                ('lr',
                                                 LogisticRegression(max_iter=
10000,
                                                                    solver='s
aga'))]),
             param_grid=[{'lr__C': [0.001, 0.01, 0.1, 1, 10.0, 100.
0, 1000.0,
                                    10000.0],
                          'lr__penalty': ['l1', 'l2'],
                          'tfidf__max_features': [5, 50, 100]}])
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

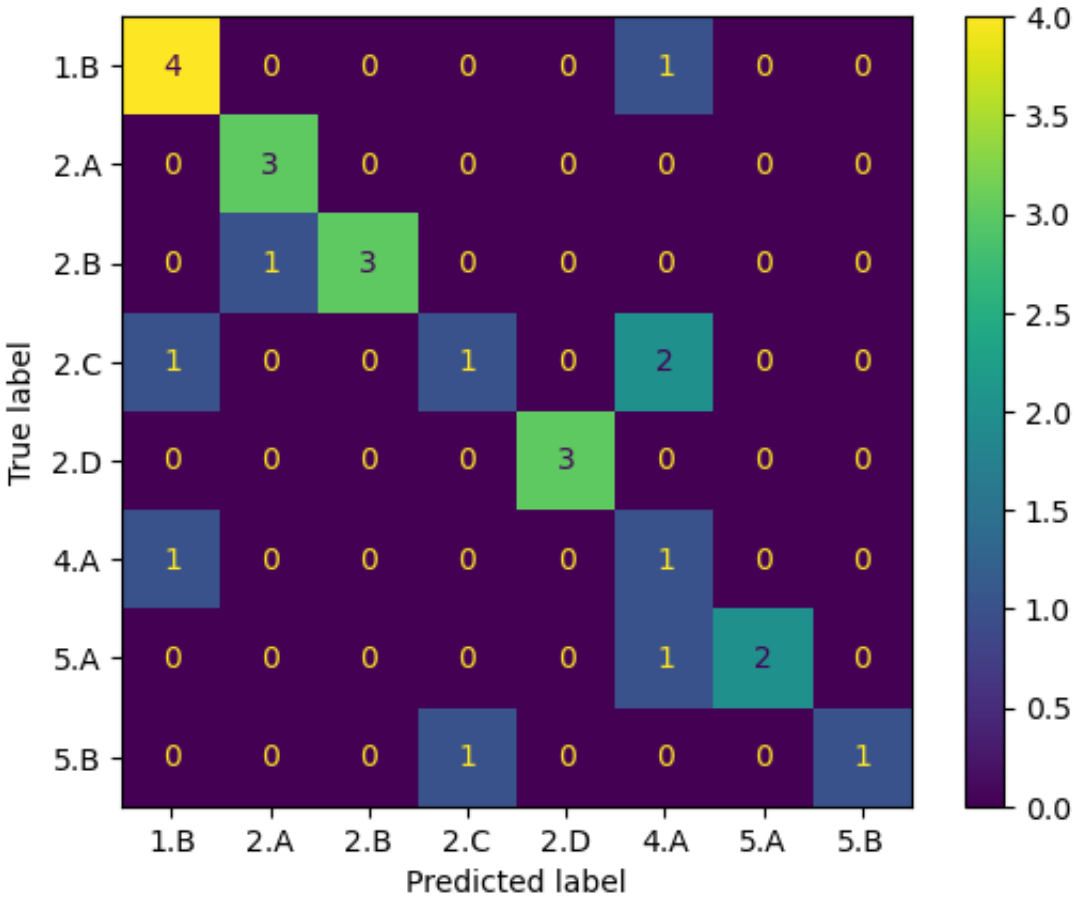
In [14]:

```
y_test_pred = gs.predict(X_test)
print(classification_report(y_test, y_test_pred))
con_mat = confusion_matrix(y_test, y_test_pred)
```

```
              precision    recall  f1-score   support

         1.B       0.67      0.80      0.73         5
         2.A       0.75      1.00      0.86         3
         2.B       1.00      0.75      0.86         4
         2.C       0.50      0.25      0.33         4
         2.D       1.00      1.00      1.00         3
         4.A       0.20      0.50      0.29         2
         5.A       1.00      0.67      0.80         3
         5.B       1.00      0.50      0.67         2

    accuracy                           0.69        26
   macro avg       0.76      0.68      0.69        26
weighted avg       0.77      0.69      0.70        26
```

In [25]:

```
cm = ConfusionMatrixDisplay(con_mat, display_labels=gs.best_estimator_.clas
cm.figure_.savefig("Reports/Images/LR_ConfusionMatrix.png")
```



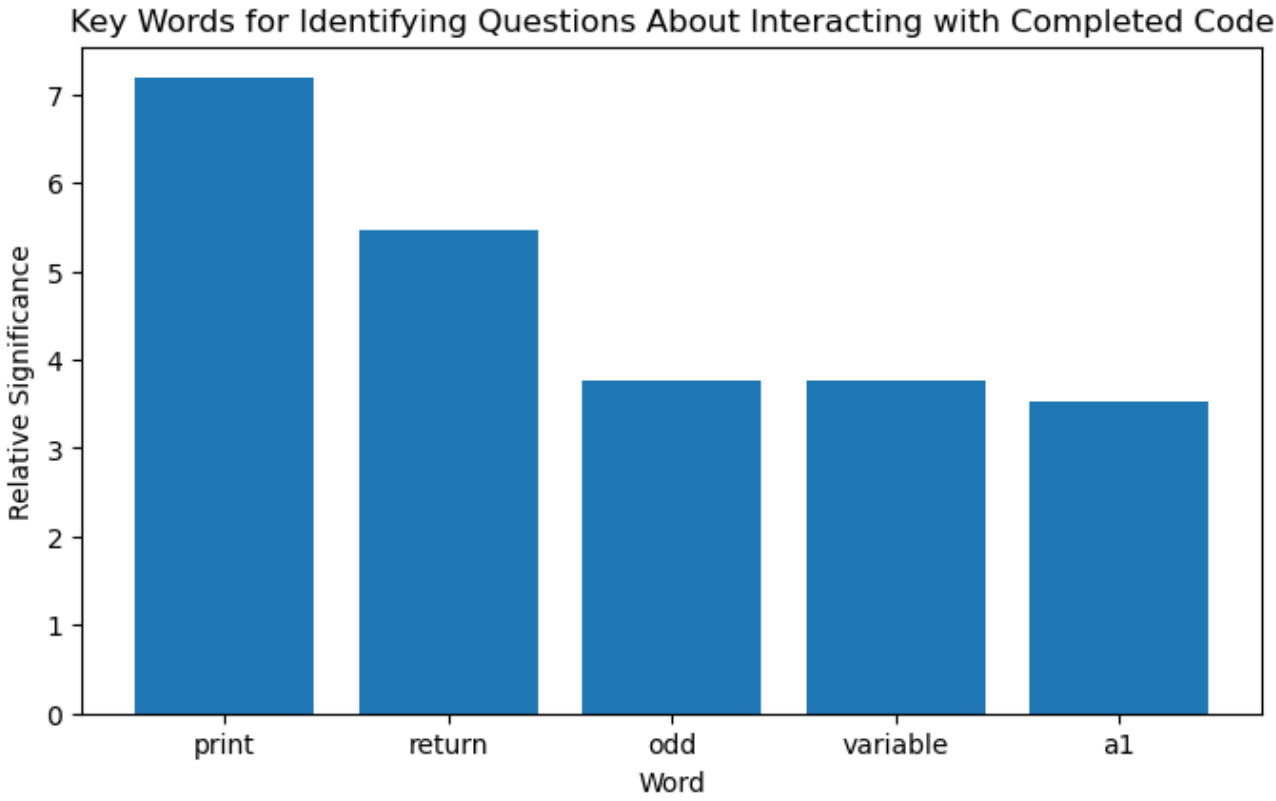
Interpreting the Logistic Classifier to Identify Key Features

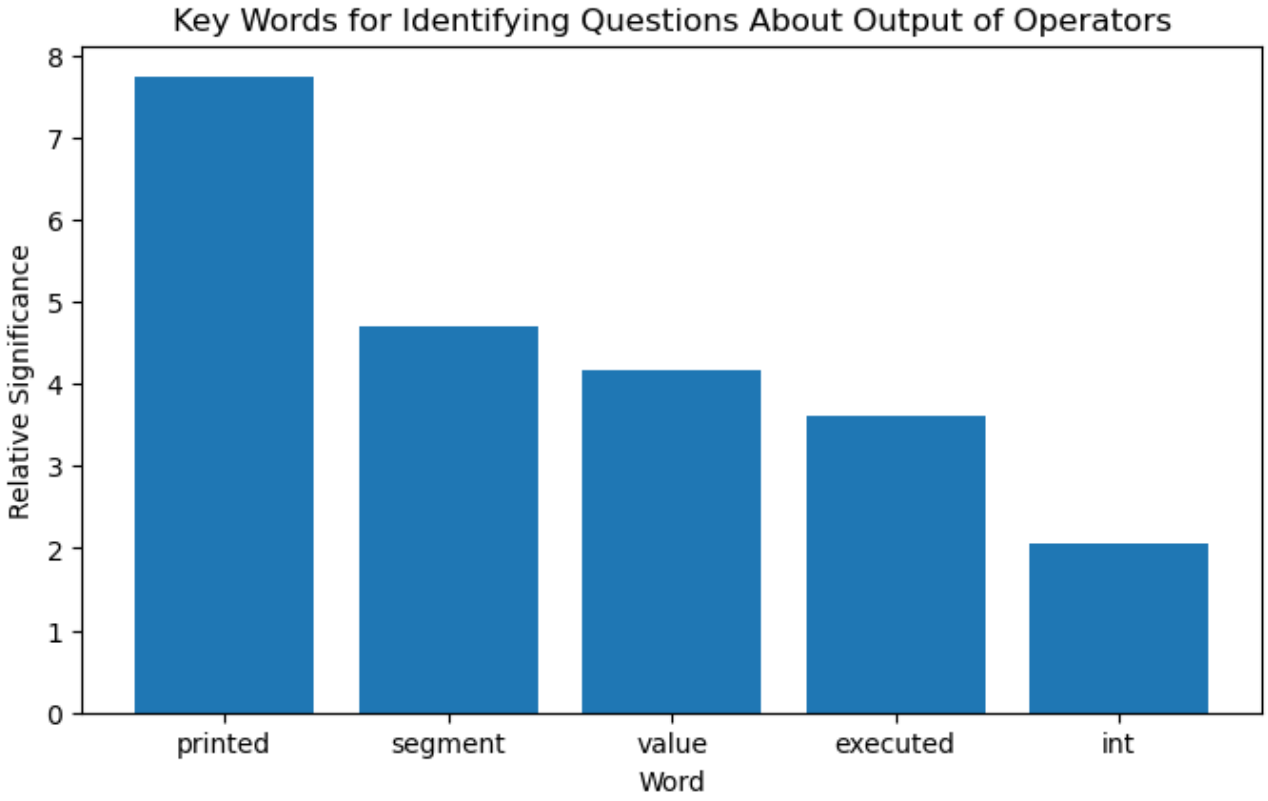
Determine the most significant works for identifying each question.

In [80]:

```
voc = {}
for x in gs.best_estimator_["tfidf"].vocabulary_:
    voc[gs.best_estimator_["tfidf"].vocabulary_[x]] = x
voc={k: v for k, v in sorted(gs.best_estimator_["tfidf"].vocabulary_.items(
#pipe["lr"].coef_[0]

key_features = list(zip(list(voc.keys()), gs.best_estimator_["lr"].coef_[0]

key_features.sort(key=lambda x: x[1], reverse=True);

fig, ax = plt.subplots(2,1, figsize=(8,10))

features = [x[0] for x in key_features[0:5]]
significance = [x[1] for x in key_features[0:5]]
ax[0].bar(features, significance)
ax[0].set_title("Key Words for Identifying Questions About Interacting with
ax[0].set_xlabel("Word");
ax[0].set_ylabel("Relative Significance");
```

```
key_features = list(zip(list(voc.keys()), gs.best_estimator_["lr"].coef_[1]
key_features.sort(key=lambda x: x[1], reverse=True);

features = [x[0] for x in key_features[0:5]]
significance = [x[1] for x in key_features[0:5]]
ax[1].bar(features, significance)
ax[1].set_title("Key Words for Identifying Questions About Output of Operat
ax[1].set_xlabel("Word");
ax[1].set_ylabel("Relative Significance");

fig.savefig("Reports/Images/InterpretTFIDF.png")
```





Using Model to Build a Question Classifier Tool

Here, a teacher can write a question, and the tool will classify the skills being assessed:

In [60]:
```
my_question = """
The method findMin(int a) below has been written to find the smallest numbe
calls to findMin will show there is an error in the code? (A) findMin(5), (
"""
results = gs.predict_proba([my_question])
results = list(zip(gs.best_estimator_.classes_, results[0]))
results.sort(key=lambda x:x[1], reverse=True)
results
```

Out[60]:
```
[('2.C', 0.6038800523277152),
 ('1.B', 0.08747366488858893),
 ('5.A', 0.07436605457702102),
 ('5.B', 0.06829595662636705),
 ('4.A', 0.061014560800013024),
 ('2.B', 0.04068396185484098),
 ('2.D', 0.038813509340998205),
 ('2.A', 0.025472239584455474)]
```

Classification with Sentence Transformer Encoding

Next, the questions will be vectorized using a Sentence Transformer Embedding, which may capture some of the semantics in the question lost in the earlier TF-IDF vectorier.

In [61]:
```
from sentence_transformers import SentenceTransformer

model = SentenceTransformer("all-MiniLM-L6-v2")
```

In [62]:
```
X_train_encode = model.encode(X_train.to_numpy())
X_test_encode = model.encode(X_test.to_numpy())
```

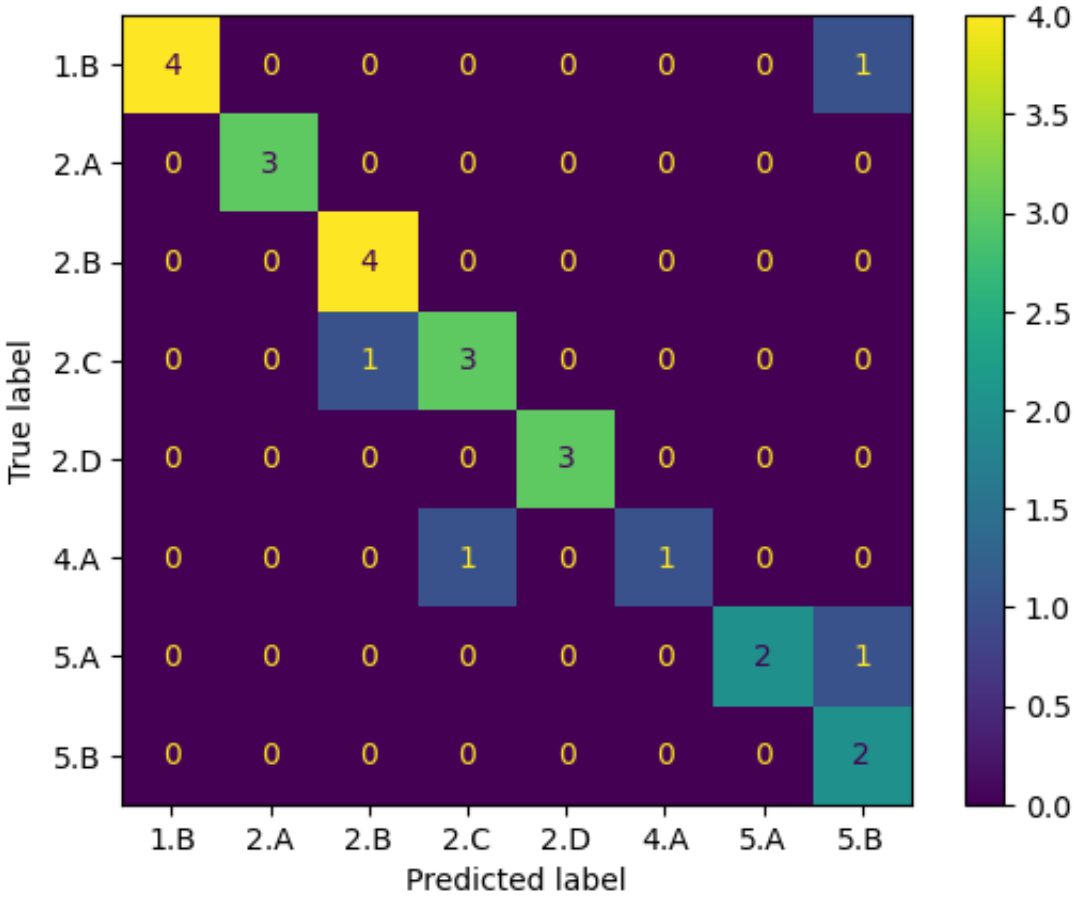
In [63]:
```
lr = LogisticRegression(solver='saga', max_iter=10000)
params = [{"penalty":['l1','l2'], 'C':[1e-3, 1e-2, 1e-1, 1, 1e1, 1e2, 1e3,
search = GridSearchCV(estimator=lr, param_grid=params)

search.fit(X_train_encode, y_train)
```

Out[63]:
```
GridSearchCV(estimator=LogisticRegression(max_iter=10000, solver='sa
ga'),
             param_grid=[{'C': [0.001, 0.01, 0.1, 1, 10.0, 100.0, 10
00.0,
                                10000.0],
                          'penalty': ['l1', 'l2']}])
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [64]:
```
y_test_pred = search.predict(X_test_encode)
```

In [81]:
```
cm = confusion_matrix(y_test, y_test_pred);
cm = ConfusionMatrixDisplay(cm, display_labels=search.best_estimator_.class

cm.figure_.savefig("Reports/Images/ST_ConfusionMatrix.png")
```



In [66]:
```
print(classification_report(y_test, y_test_pred))
```

```
              precision    recall  f1-score   support

         1.B       1.00      0.80      0.89         5
         2.A       1.00      1.00      1.00         3
         2.B       0.80      1.00      0.89         4
         2.C       0.75      0.75      0.75         4
         2.D       1.00      1.00      1.00         3
         4.A       1.00      0.50      0.67         2
         5.A       1.00      0.67      0.80         3
         5.B       0.50      1.00      0.67         2

    accuracy                           0.85        26
   macro avg       0.88      0.84      0.83        26
weighted avg       0.89      0.85      0.85        26
```

In [67]:

```
my_question = """
The method findMin(int a) below has been written to find the smallest numbe
calls to findMin will show there is an error in the code? (A) findMin(5), (
"""
results = search.predict_proba([model.encode(my_question)])
results = list(zip(search.best_estimator_.classes_, results[0]))
results.sort(key=lambda x:x[1], reverse=True)
results
```

Out[67]:

```
[('4.A', 0.8061548),
 ('1.B', 0.086053886),
 ('5.B', 0.068511195),
 ('2.C', 0.024228672),
 ('5.A', 0.009655114),
 ('2.B', 0.0029592994),
 ('2.A', 0.0013785751),
 ('2.D', 0.001058582)]
```

Build and Test a ChatGPT Classifier

This classifier asks ChatGPT to determine the Computational Thinking Skill being assessed in the problem.

1. Create a function call to ask ChatGPT for the classification
2. Test ChatGPT on data set and evaluate classification.

In [70]:

```
import pandas as pd
from openai import OpenAI
import os

from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
```

```
from sklearn.metrics import ConfusionMatrixDisplay
```

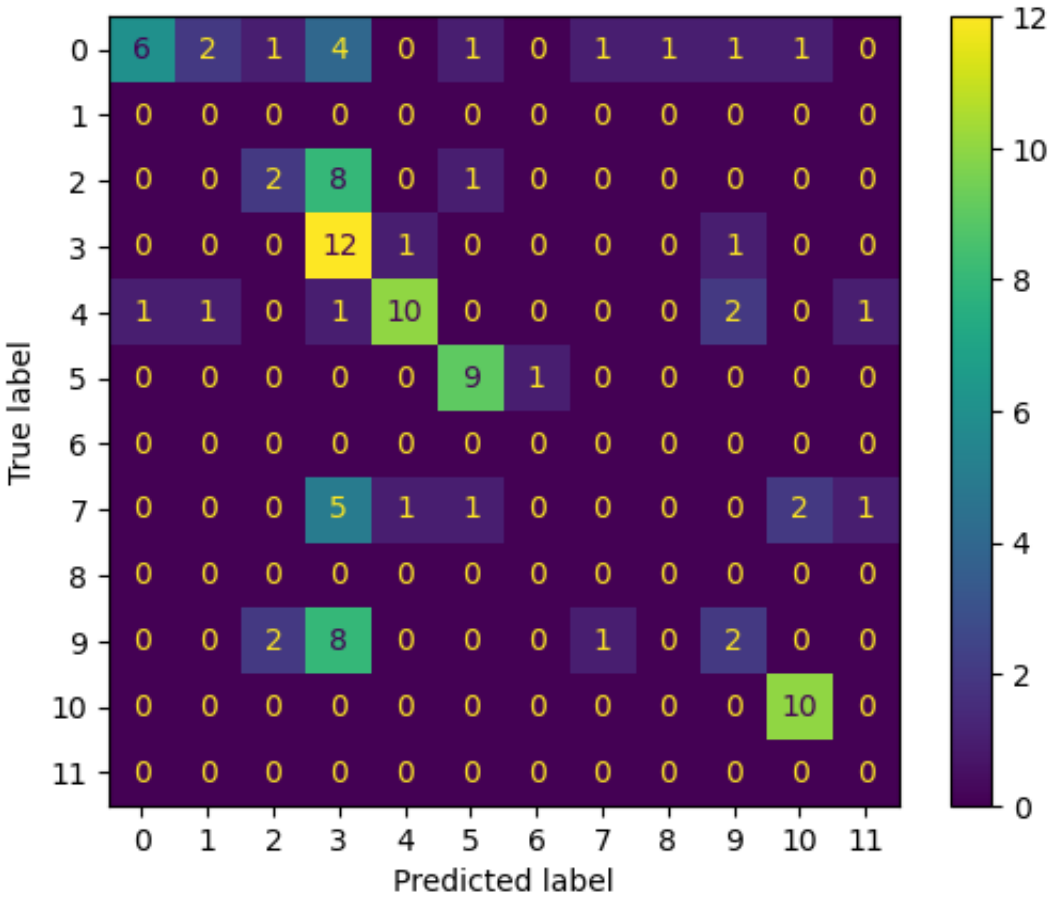
In [71]:

```
prompt_start = """
Here are the categories for AP questions.
1.B: Determine code that would be used to complete code segments
1.C: Determine code that would be used to interact with completed program c
2.A: Apply the meaning of specific operators
2.B: Determine the result or output based on statement execution order in a
2.C: Determine the result or output based on the statement execution order
2.D: Determine the number of times a code segment will execute.
4.A: Use test-cases to find errors or validate results.
4.B: Identify errors in program code.
4.C: Determine if two or more code segments yield equivalent results.
5.A: Determine the behavior of a given segment of program code.
5.B: Explain why a code segment will not compile or work as intended
5.C: Explain how the result of program code changes, given a change to the
5.D: Describe the initial conditions that must be met for a program segment

Below are three example questions for each category
1.B: Consider the mode method, which is intended to return the most
1.B: Which of the following code segments produces the output "9876543
1.B: Consider the following method which is intended to print the va
1.C: Consider the following class definition. public class Value { p
1.C: Which of the following statements assigns a random integer between 25
1.C: Consider the following class definition. public class Example{ private
2.A: Consider the following code segment. Assume num is a properly de
2.A: Consider the following code segment. int a=5; int b=2; double c=3.0;
2.A: Consider the following code segment. int x = 7; int y = 3; if ((x <
2.B: Consider the following code segment. int[][] values = {{1, 2, 3}
2.B: Consider the following code segment. int[] arr = {7 2 5 3 0 10};
2.B: Consider the following code segment. int[] arr = {1 2 3 4 5 6 7}
2.C: Consider the following code segment. ArrayList<Integer> numList =
2.C: Consider the following method. public static int mystery(int[] arr) {
2.C: Consider the following instance variable and method. private List<Stri
2.D: Consider the following code segment. String[] testTwo = {"last "
2.D: Consider the following code segment. Assume num is a properly de
4.A: Consider the following method. public static void printSome(int
4.A: Vehicles are classified based on their total interior volume. The clas
4.A: Consider the following recursive method. public static boolean
4.C: Consider the following code segment. int num = /* initial value n
4.C: Consider the following class declaration. public class Student { pri
4.C: For which of the following test cases will the call seqSearchRec(5) a
5.A: Consider the following code segment. Assume num is a properly de
5.A: Consider the following code segment. Assume that a is greater th
5.A: Consider the following code segment. Assume that num3 > num2 >
5.B: The following method is intended to remove all elements of an Arr
5.B: Consider the following Book and AudioBook classes. public class Book
5.B: Consider the following method, which is intended to return a list cont
5.D: Consider the following statement. Assume that a and b are properly
5.D: Consider the following methods. /** Precondition: a>0 and b > 0 */
Given the categories and examples above, which class does the following que
"""
```

In [72]:
```
client = OpenAI(api_key=os.environ.get("jeff_api"))

def gpt_classify(prompt):
  response = client.chat.completions.create(
  model="gpt-4o-mini",
  messages=[{"role": "user", "content": prompt_start + prompt}]
  )
  return response.choices[0].message.content
```

In [73]:
```
df["GPT_Classify"] = df["Question"].apply(gpt_classify)
```

In [75]:
```
con_mat = confusion_matrix(df["Classification"], df["GPT_Classify"])
cm = ConfusionMatrixDisplay(confusion_matrix=con_mat).plot()
cm.figure_.savefig("Reports/Images/ChatGPT_ConfusionMatrix.png")
```



Using ChatGPT Model to Build a Question Classifier

Here, a teacher could enter a sample question, and ChatGPT will identify the

Computational Thinking Skill being assessed.

In [79]:
```
my_question = """
What will be the output of the following code? int a=5; if a%5==0 System.ou
"""
prompt_part2 = """For each possible class, please give the probability the
(1.B, 0.92), (2.A, 0.04), (2.B, 0.01), (2.C, 0.01), (2.D, 0.00), (4.A, 0.01
gpt_classify(prompt_part2 + my_question)
```

Out[79]:
```
'(2.B, 0.91), (5.B, 0.05), (1.B, 0.03), (4.A, 0.01), (5.A, 0.00), (2.A, 0.0
0), (2.C, 0.00), (2.D, 0.00)'
```

Summary

This project had some success at identifying the skills assessed in a particular question.

1. The classifier using a TF-IDF vectorization accurately identified the Computational Thinking Practice being accessed in 70% of the questions, while the simple classifier using the Sentence Transformed identified the Computational Thinking Practice being assessed in 85% of the questions.
2. ChatGPT had an accuracy of 48% in identifying the Computational Thinking Practice assessed in a question.
3. A simple tool was developed for identifying the skill assessed by a question

Next Steps:

1. Develop better prompts to improve ChatGPT's classification accuracy.
2. Determine whether the classifier can also identify the "Essential Knowledge" assessed by the question, not just the computational skill.
3. Attempt to generalize the classifiers to classify non-assessment questions such as lecture material, lab questions, and homework problems.
4. Create a visualization that shows the distribution of thinking skills and content assessed over the course of the curriculum.