A DATASHEET FOR DATASET

This datasheet is anonymized for submission. Responses in *italics* below will be expanded upon publication.

A.1 MOTIVATION

For what purpose was the dataset created? Was there a specific task in mind? Was there a specific gap that needed to be filled? This dataset was created as a benchmark for code generation models. It was created with the goal of filling two gaps in the existing benchmarks: 1) the need for a benchmark with multiple natural language descriptions per problem and 2) the need for a benchmark targeting a specific programmer skill level.

Who created the dataset and on behalf of which entity? *This dataset was created by a set of anonymized researchers.*

Who funded the creation of the dataset? *This work received external funding.*

A.2 COMPOSITION

What do the instances that comprise the dataset represent? Are there multiple types of instances? The dataset consists of programming tasks where each task has the following components: a function signature; a test suite; an expert-written implementation; an expert-written prompt; and a set of student-written prompts (minimum 14).

How many instances are there in total? There are 48 programming tasks and 1,749 student-written prompts.

Does the dataset contain all possible instances or is it a sample of instances from a larger set? The dataset is not a sample of a larger set, but it does not contain all possible instances, since additional programming tasks or prompts could be devised.

What data does each instance consist of? "Raw" data or features? Each instance consists of text. The dataset as a whole is stored in CSV format.

Is there a label or target associated with each instance? Each prompt is labeled with the name of the programming task it is associated with.

Is any information missing from individual instances? If so, please provide a description, explaining why this information is missing. This does not include intentionally removed information, but might include, e.g., redacted text. We have intentionally de-identified the dataset so that prompts cannot be traced back to the students who wrote them.

Are relationships between individual instances made explicit? If so, please describe how these relationships are made explicit. Yes. Each prompt is associated with the programming task it described, and we have also provided an ID linking together prompts written by the same student.

Are there recommended data splits? If so, please provide a description of these splits, explaining the rationale behind them. No.

Are there any errors, sources of noise, or redundancies in the dataset? Some prompts may be appear multiple times in the dataset, either because multiple students described the problem in the same way, or because a student re-submitted a prompt without editing it.

Is the dataset self-contained, or does it link to or otherwise rely on external resources? The dataset is self-contained and does not link to or rely on external resources.

Does the dataset contain data that might be considered confidential? No.

Does the dataset contain data that, if viewed directly, might be offensive, insulting, threatening, or might otherwise cause anxiety? No.

Does the dataset relate to people? Yes.

Does the dataset identify any subpopulations? No.

Is it possible to identify individuals, either directly or indirectly from the dataset? If so, please describe how. We believe that this is not possible. We have de-identified the dataset, removing participant usernames and replacing them with randomly generated numeric IDs. We have also manually reviewed the dataset to ensure that there is no personally-identifying information that could link prompts back to participants.

Does the dataset contain data that might be considered sensitive in any way? No.

A.3 COLLECTION PROCESS

How was the data associated with each instance acquired? Was the data directly observable, reported by subjects, or indirectly inferred/derived from other data? If data was reported by subjects or indirectly inferred/derived from other data, was the data validated/verified? The data was collected during a human-subjects experiment, where participants wrote prompts to describe a function. Participants were then presented with a solution generated by the Codex code generation model (Chen et al., 2021), as well as the results of running a suite of test cases on the solution. Participants could submit any number of prompts for a particular problem (within the time limit of the experiment).

Participants submitted the data using our web-based experiment platform. There was no validation of their submissions, but the experiment was overseen in real time by an experimenter, who was available to answer questions and intervene when participants ran into issues with the task.

What mechanisms or procedures were used to collect the data? How were these mechanisms or procedures validated? The experiment was conducted on a web-based experiment platform built by the research team. We conducted a small-scale pilot study to assess the functionality of the platform.

If the dataset is a sampled from a larger set, what was the sampling strategy? The dataset is not sampled.

Who was involved in the data collection process and how were they compensated? *We recruited participants who had taken an introductory Python programming course at our institutions within the past two years. Each participant received a $50 gift card for participating in the approximately 75 minute task.*

Over what timeframe was the data collected? Does this timeframe match the creation timeframe of the data associated with the instances? The data was collected between November 2022 and May 2023.

Were any ethical review processes conducted? If so, please provide a description of these review processes, including the outcomes, as well as a link or other access point to any supporting documentation. *The study was conducted under supervision of a single Institutional Review Board with authorization agreements as needed.*

Does the dataset relate to people? Yes.

Did you collect the data from the individuals in question directly, or obtain it via third parties or other sources? We collected the data directly from participants as part of a lab-based study.

Were the individuals in question notified about the data collection? If so, please describe how notice was provided, and provide a link or other access point to, or otherwise reproduce, the

exact language of the notification itself. Participants submitted an informed consent form prior to participating. They verbally affirmed their ongoing consent at the beginning of the study.

Did the individuals in question consent to the collection and use of their data? If so, please describe how consent was requested and provided, and provide a link or other access point to, or otherwise reproduce, the exact language to which the individuals consented. *The consent form will be released publicly upon acceptance, as it is not anonymized.*

If consent was obtained, were the consenting individuals provided with a mechanism to revoke their consent in the future or for certain uses? If so, please provide a description, as well as a link or other access point to the mechanism. Participants were allowed to retract their data prior to its public release, by contacting the researchers.

Has an analysis of the potential impact of the dataset and its use on data subjects been conducted? Yes, the impact of releasing this data was considering during the IRB process.

A.4 PREPROCESSING/CLEANING/LABELING

Was any preprocessing/cleaning/labeling of the data done? If so, please provide a description. If not, you may skip the remainder of the questions in this section. The dataset was de-identified to preserve participant anonymity; we removed all usernames and replaced them with randomly generated numeric IDs.

Tokenization was done to facilitate the analysis presented in the paper, but the released dataset contains the submitted responses, not the tokenized ones. The tokenization script *will be made available upon publication.*

Was the "raw" data saved in addition to the preprocessed/cleaned/labeled data? If so, please provide a link or other access point to the "raw" data. We will only release the de-identified dataset, not the raw dataset. This is in order to preserve participant anonymity.

Is the software used to preprocess/clean/label the instances available? Yes. *All code will be made available upon publication.*

A.5 USES

Has the dataset been used for any tasks already? No.

Is there a repository that links to any or all papers or systems that use the dataset? No.

What (other) tasks could the dataset be used for? The primary intended use for this dataset is as a benchmark. However, it could also be used to fine-tune machine learning models. We imagine that it could be useful to fine-tune a code generation model to better handle the way students talk about code. It could also be used in tandem with pass@k rates to fine-tune a prompt classification model.

Is there anything about the composition of the dataset or the way it was collected and preprocessed/cleaned/labeled that might impact future uses? For example, is there anything that a future user might need to know to avoid uses that could result in unfair treatment of individuals or groups or other undesirable harms. If so, please provide a description. Is there anything a future user could do to mitigate these undesirable harms? Although a diverse population of students was involved in this research, representing many dialects of English, we have not examined how representative it is of the population of English-speaking students. As a result, benchmark results calculated with this dataset might not generalize to all populations of potential code generation users.

Are there tasks for which the dataset should not be used? If so, please provide a description. This dataset should not be used to build artificial intelligence that aims to deceive humans (e.g. by spreading misinformation or by impersonating a human). Certain uses are also restricted by the dataset's OpenRAIL license.

A.6 DISTRIBUTION

Will the dataset be distributed to third parties outside of the entity on behalf of which the dataset was created? The de-identified dataset will be made public.

How will the dataset will be distributed? Does the dataset have a digital object identifier (DOI)? *The dataset will be distributed through the Open Science Framework once dataset is made public.*

When will the dataset be distributed? *The dataset will be distributed via the Open Science Framework.*

Will the dataset be distributed under a copyright or other intellectual property (IP) license, and/or under applicable terms of use (ToU)? If so, please describe this license and/or ToU, and provide a link or other access point to, or otherwise reproduce, any relevant licensing terms or ToU, as well as any fees associated with these restrictions. The dataset is licensed under an OpenRAIL-D License Agreement.

Have any third parties imposed IP-based or other restrictions on the data associated with the instances? If so, please describe these restrictions, and provide a link or other access point to, or otherwise reproduce, any relevant licensing terms, as well as any fees associated with these restrictions. No.

Do any export controls or other regulatory restrictions apply to the dataset or to individual instances? No.

A.7 MAINTENANCE

Who will be supporting/hosting/maintaining the dataset? The Open Science Framework will host the dataset.

How can the owner/curator/manager of the dataset be contacted (e.g., email address)? *Questions or concerns about this dataset can be directed to the authors.*

Is there an erratum? No, but if errors are discovered, we will post one to the Open Science Framework project.

Will the dataset be updated? No. The dataset is stable.

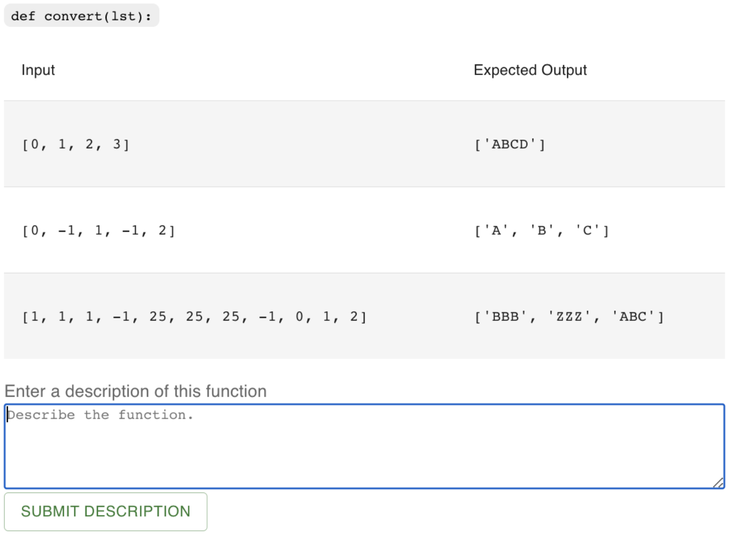
If the dataset relates to people, are there applicable limits on the retention of the data associated with the instances? If so, please describe these limits and explain how they will be enforced. Participants were informed that their de-identified data would be published online and remain accessible in perpetuity. All identifying information will be destroyed, as outlined in the IRB protocol.

Will older versions of the dataset continue to be supported/hosted/maintained? We do not plan to update this dataset.

If others want to extend/augment/build on/contribute to the dataset, is there a mechanism for them to do so? If so, will these contributions be validated/verified? If not, why not? We welcome replications of our work, but we think these should be published as separate datasets in order to make the population and data collection times clear. Students are a dynamic population, and their experience with writing prompts is likely to change in the near future.

B DETAILS OF WEB APPLICATION FOR DATA COLLECTION

We collected data via a purpose-built web-based application. See Figures 8, 9, and 10 for screenshots of the interface. Students encountered 8 different problems selected from the 48 overall as part of the study. The interface remained the same for each problem.



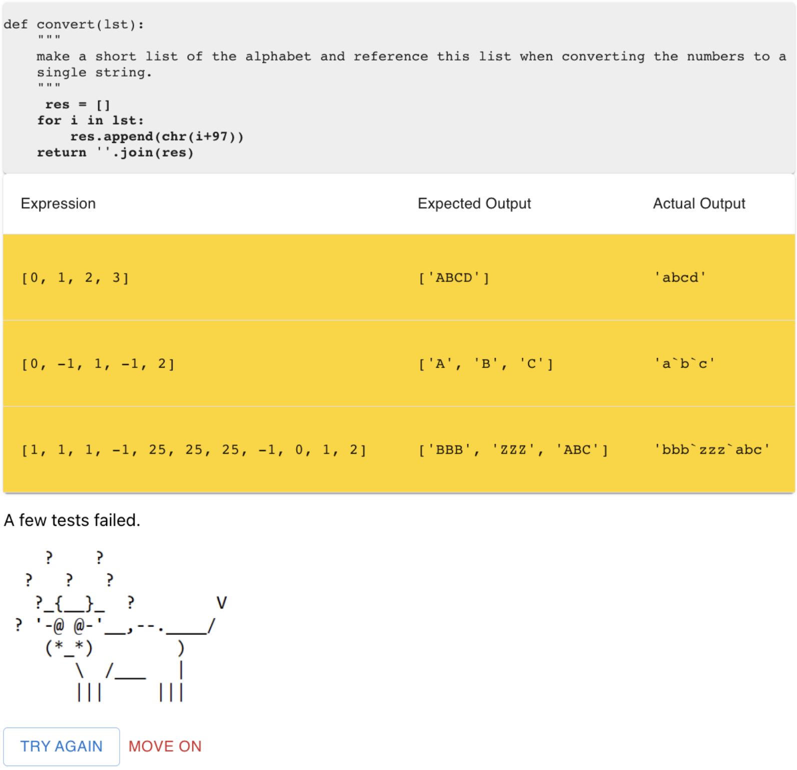
```
def convert(lst):
```

Input	Expected Output
`[0, 1, 2, 3]`	`['ABCD']`
`[0, -1, 1, -1, 2]`	`['A', 'B', 'C']`
`[1, 1, 1, -1, 25, 25, 25, -1, 0, 1, 2]`	`['BBB', 'ZZZ', 'ABC']`

Enter a description of this function

Describe the function.

Figure 8: The screen that students see when entering a problem description.



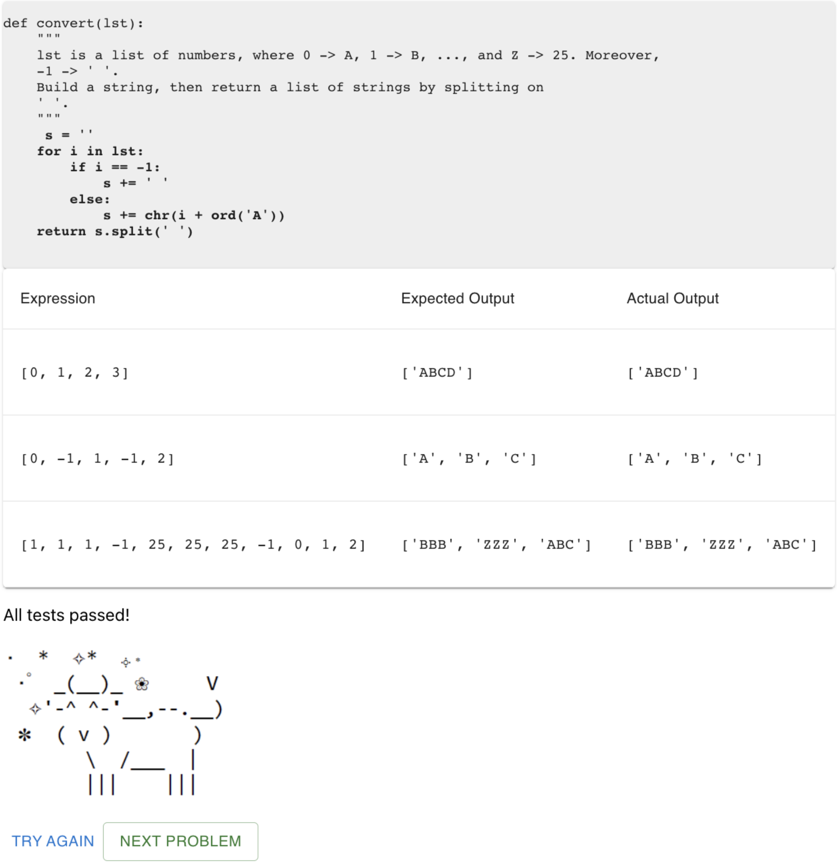
```
def convert(lst):
    """
    make a short list of the alphabet and reference this list when converting the numbers to a
    single string.
    """
     res = []
    for i in lst:
        res.append(chr(i+97))
    return ''.join(res)
```

Expression	Expected Output	Actual Output
`[0, 1, 2, 3]`	`['ABCD']`	`'abcd'`
`[0, -1, 1, -1, 2]`	`['A', 'B', 'C']`	``'a`b`c'``
`[1, 1, 1, -1, 25, 25, 25, -1, 0, 1, 2]`	`['BBB', 'ZZZ', 'ABC']`	``'bbb`zzz`abc'``

A few tests failed.

```
      ?    ?
   ?   ?   ?
     ?_{__}_  ?        V
  ?  '-@ @-'__,--.____/
      (*_*)        )
         \  /___   |
          |||      |||
```

Figure 9: We run expert tests automatically and highlight ones they fail.



```
def convert(lst):
    """
    lst is a list of numbers, where 0 -> A, 1 -> B, ..., and Z -> 25. Moreover,
    -1 -> ' '.
    Build a string, then return a list of strings by splitting on
    ' '.
    """
     s = ''
    for i in lst:
        if i == -1:
            s += ' '
        else:
            s += chr(i + ord('A'))
    return s.split(' ')
```

Expression	Expected Output	Actual Output
`[0, 1, 2, 3]`	`['ABCD']`	`['ABCD']`
`[0, -1, 1, -1, 2]`	`['A', 'B', 'C']`	`['A', 'B', 'C']`
`[1, 1, 1, -1, 25, 25, 25, -1, 0, 1, 2]`	`['BBB', 'ZZZ', 'ABC']`	`['BBB', 'ZZZ', 'ABC']`

All tests passed!

Figure 10: The screen students see when all tests pass.

B.1 ELIGIBILITY CRITERIA

STUDENTEVAL is built from problems selected from three undergraduate, introduction to computing courses. All three courses are taught using Python and cover the same basic topics.

Students were eligibility for the study if they were currently an undergraduate student at one of the three institutions and had completed one of the above courses over five possible terms (Fall 2021 through Spring 2023). Students could be currently enrolled in a subsequent course, but were not eligible for the study if they had completed subsequent courses.

B.2 TUTORIAL/TRAINING INFORMATION

Instructions for how study participants should interact with the Code LLM were provided via three tutorial problems. Each problem was chosen specifically to exhibit the range of possible Code LLM interactions. We provided a working prompt alongside the first problem (to showcase model success), the second problem was an "impossible" problem (to showcase model failure), and the third was an easy-to-solve problem that modeled the full participant interaction. Relevant text describing each tutorial problem is provided below:

- **Problem 1:** "The inputs and expected outputs are examples of what the function should do, and the text box is where you'll describe the function's behavior. Try copy and pasting: "Takes in a string, and returns an integer representing the maximum number that the same letter appears consecutively in the string." then click SUBMIT DESCRIPTION."
- **Problem 2:** "Looking at the inputs and expected outputs below, try to figure out what the function `apply_operations` should do, and type it in the text box below."
- **Problem 3:** "Try writing a description for this function on your own based on the inputs and expected outputs, then click SUBMIT DESCRIPTION."

B.3 STUDY TIMING INFORMATION

Participants completed the tutorial, provided descriptions, and performed an exit survey & interview over approximately 75 minutes. During the main experiment, students were randomly assigned eight problems out of the total 48 problems in STUDENTEVAL. The first four problems were untimed and students had, at maximum, 5 minutes to work on each of the second set of four problems. Throughout the study, students were given the option to "Try Again" or "Move On" to the next problem at will. This is one of the causes of the variation in the number of responses per user, per problem in STUDENTEVAL.

C DATASET DETAILS

C.1 DATASET EXAMPLES

To illustrate the variety of student-written descriptions, we show three descriptions of the `total_bill` problem that are in the *Last Success* subset:

1. The function takes in some number of lists consisting of a string, an integer, and a number, as well as one additional number "sales tax". The function multiplies the integer and the number in each list and adds them together for all the lists, giving a "total". Then the function multiplies the "total" by the "sales tax" and outputs that value added to the "total", truncated to two decimal places.
2. Add up the values of the product of the values stored in index 1 and 2 and round to the nearest hundredths if there are more than 2 decimal places.
3. you will have two inputs a list of lists and the tax rate. for every list in the list of lists multiply the second and third item and add all of them and then multiply that by the sales tax plus 1. if the resulting number has more than two decimal places shorten it to two decimal places.

And three descriptions of the same problem in the *Last Failure* subset:

1. This function takes in a list of the item purchased, the amount of the item purchased, the price for each item, and the overall sales tax. The amount purchased is multiplied with the price for each item, creating a total amount. The sales tax is then multiplied by the outcome of the total amount, and then the result of the multiplication is added onto the total price. The total price is then returned as the output.
2. takes a list of groceries and a value for sales tax. the list of groceries contains a list for every item. each item's list contains its name, quantity, and price, in that order. returns an float that is the sales tax times times the sum of all goods' quantity*price
3. Enter a list where the 0th index is a list of lists and the 1st index is an integer. for every list in the 0th index, for every list multiply the integers in the first and second indexes of each list in the list of lists. Add each product of each list. Then, multiply by 1 plus the integer in the first index of the input list

C.2 PROMPT RELIABILITY

Table 2 shows the number of reliable and unreliable prompts in each subset for all benchmarked models.

D PROMPT ANALYSIS DETAILS

D.1 TOKENIZATION & PRE-PROCESSING

As a pre-processing step, we replace functionally equivalent words with placeholders. This pre-processing was done as, across all problems in STUDENTEVAL, there are 48 different function names (e.g., `convert`, `fib`) and 57 different argument names (e.g., `val`, `meetings`). Therefore, we replace references to functions and parameters with *FUNCTIONNAME* and *PARAM*, respectively. Our approach does not handle capital function or argument names, as their meaning is ambiguous (e.g., "Convert" is a verb, but `convert` is a function name). We also replace "return"/"returns" with *RETURN*.

Student prompts consist of a mix of Python terminology (including code snippets) and English words. Therefore, standard English tokenization libraries were insufficient. We perform a best-effort tokenization using a regex-based Python function that performs multiple passes. The overall goal was to maintain meaningful code-related items as single terms. Specifically, we treat list indexing, lists, dictionaries, single/double quote strings, numbers, and comparison operators as single tokens. There may be possessives and/or contractions that are tokenized as strings rather than separate terms in the dataset. Terms were additionally lowercased and basic stopwords which are not meaningful in a programming context were filtered out.

D.2 TF-IDF ANALYSIS

We used `scikitlearn`'s `TfidfVectorizer` with our tokenizer to generate the TF-IDF values presented in this paper. Figure 11 presents the mean scores for the top 25 words for each of the four subsets individually.

D.3 REGRESSION ANALYSIS

We fitted mixed-effects regression models to predict STUDENTEVAL pass@1 rates estimated with completions obtained from StarCoderBase. Models were fitted using the `lme4` library in R. All models included random intercepts for problems; random slopes were omitted due to model complexity. For vocabulary-level features, we use indicator variables: a 1 if the prompt uses the word and a 0 otherwise. Values that are statistically significant with a threshold of $p = 0.5$ are displayed in **bold**.

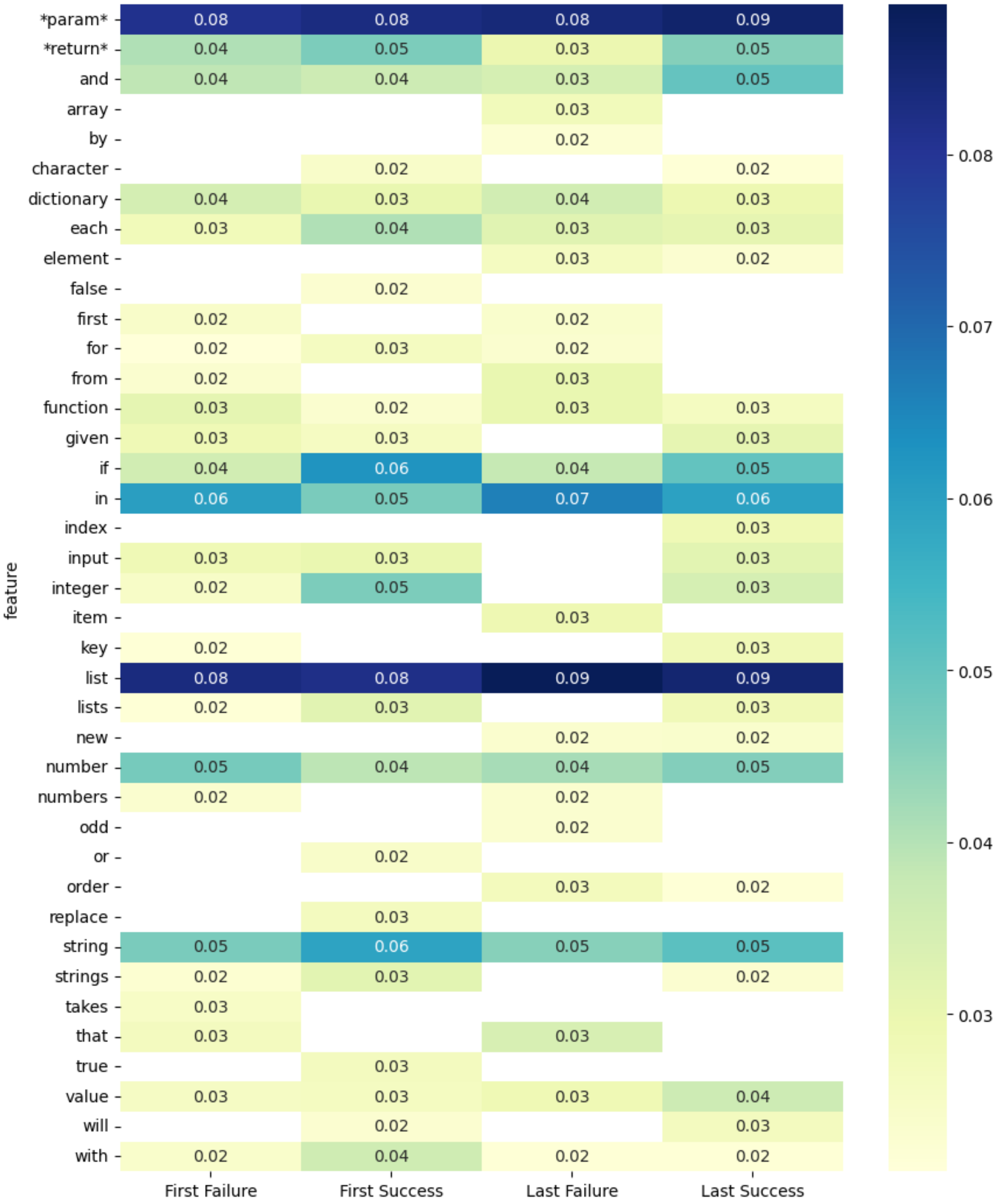
Prompt length Prompt length was calculated by number of tokens using the tokenizer discussed in Section D.1. The raw token count was divided by 100 for scaling purposes. The full estimates are shown in Table 3.

Table 2: Number of reliable and unreliable prompts by model and subset.

Model	Rate Subset	pass@1 < 0.2	pass@1 > 0.8
Code-Llama-Py-13B	failure (first attempt)	38	-
	failure (last attempt)	34	1
	success (first attempt)	6	19
	success (last attempt)	3	12
Code-Llama-Py-34B	failure (first attempt)	36	-
	failure (last attempt)	34	3
	success (first attempt)	5	16
	success (last attempt)	-	13
Code-Llama-Py-7B	failure (first attempt)	41	-
	failure (last attempt)	36	1
	success (first attempt)	8	15
	success (last attempt)	3	10
GPT-3.5-Turbo-0301	failure (first attempt)	42	-
	failure (last attempt)	31	2
	success (first attempt)	9	6
	success (last attempt)	5	7
Phi-1	failure (first attempt)	35	-
	failure (last attempt)	37	-
	success (first attempt)	7	7
	success (last attempt)	13	4
Replit-Code-v1	failure (first attempt)	43	-
	failure (last attempt)	40	-
	success (first attempt)	18	2
	success (last attempt)	25	-
SantaCoder	failure (first attempt)	45	-
	failure (last attempt)	42	-
	success (first attempt)	21	3
	success (last attempt)	23	1
StarChat-Alpha	failure (first attempt)	38	-
	failure (last attempt)	37	-
	success (first attempt)	7	11
	success (last attempt)	5	6
StarCoderBase	failure (first attempt)	41	-
	failure (last attempt)	37	1
	success (first attempt)	8	12
	success (last attempt)	4	7
StarCoderBase-1B	failure (first attempt)	45	-
	failure (last attempt)	41	1
	success (first attempt)	26	2
	success (last attempt)	32	1
StarCoderBase-3B	failure (first attempt)	40	-
	failure (last attempt)	38	-
	success (first attempt)	15	8
	success (last attempt)	17	3
StarCoderBase-7B	failure (first attempt)	44	-
	failure (last attempt)	39	1
	success (first attempt)	9	9
	success (last attempt)	10	8

Fixed effects	$\widehat{\beta}$	z	p
Intercept	0.14 (+/-0.04)	3.4	**0.002**
totalLength	0.06 (+/- 0.02)	2.9	**0.007**

Table 3: Mixed-effects regression results for problem length



feature	First Failure	First Success	Last Failure	Last Success
param	0.08	0.08	0.08	0.09
return	0.04	0.05	0.03	0.05
and	0.04	0.04	0.03	0.05
array			0.03	
by			0.02	
character		0.02		0.02
dictionary	0.04	0.03	0.04	0.03
each	0.03	0.04	0.03	0.03
element			0.03	0.02
false		0.02		
first	0.02		0.02	
for	0.02	0.03	0.02	
from	0.02		0.03	
function	0.03	0.02	0.03	0.03
given	0.03	0.03		0.03
if	0.04	0.06	0.04	0.05
in	0.06	0.05	0.07	0.06
index				0.03
input	0.03	0.03		0.03
integer	0.02	0.05		0.03
item			0.03	
key	0.02			0.03
list	0.08	0.08	0.09	0.09
lists	0.02	0.03		0.03
new			0.02	0.02
number	0.05	0.04	0.04	0.05
numbers	0.02		0.02	
odd			0.02	
or		0.02		
order			0.03	0.02
replace		0.03		
string	0.05	0.06	0.05	0.05
strings	0.02	0.03		0.02
takes	0.03			
that	0.03		0.03	
true		0.03		
value	0.03	0.03	0.03	0.04
will		0.02		0.03
with	0.02	0.04	0.02	0.02

Figure 11: TF-IDF values for top 25 words in each prompt subset category.

Input/output wording We explored the impact of mentioning “return”, “input”, “print”, and “output”. We counted all stemmed mentions. The full estimates are shown in Table 4.

Fixed effects	$\widehat{\beta}$	z	p
Intercept	0.19 (+/- 0.03)	6.1	< *0.0001*
returnInd	0.067 (+/- 0.02)	4.2	< *0.0001*
inputInd	0.023 (+/- 0.03)	0.8	0.42
printInd	-0.008 (+/- 0.03)	-0.3	0.77
outputInd	0.026 (+/- 0.02)	1.2	0.23

Table 4: Mixed-effects regression results for input/output terms

Datatype mentions We explored the impact of mentioning “list”, “dictionary”, “array”, “variable”, “number”, “int”, as well as giving example lists and dictionaries (indicated by use of square or curly braces). The full estimates are shown in Table 5.

Fixed effects	$\widehat{\beta}$	z	p
Intercept	0.21 (+/- 0.03)	6.8	< *0.0001*
listInd	0.041 (+/- 0.02)	2.3	**0.02**
dictInd	0.005 (+/- 0.05)	0.12	0.90
squareBraceInd	-0.22 (+/- 0.4)	-0.6	0.55
curlyBraceInd	0.38 (+/- 0.2)	1.8	0.07
arrayInd	-0.07 (+/- 0.04)	-1.9	0.053
variableInd	0.031 (+/- 0.04)	0.8	0.41
numberInd	0.002 (+/- 0.02)	0.11	0.91
intInd	0.026 (+/- 0.02)	1.4	0.17

Table 5: Mixed-effects regression results for datatype mentions

Function and parameter names We explored the effect of mentioning the function name and the name of parameters. The full estimates are shown in Table 6.

Fixed effects	$\widehat{\beta}$	z	p
Intercept	0.24 (+/- 0.03)	8.2	< *0.0001*
paramInd	0.01 (+/- 0.02)	0.6	0.57
functionnameInd	-0.08 (+/- 0.03)	-2.4	*0.02*

Table 6: Mixed-effects regression results for function name and parameter name mentions

D.4 EMBEDDING PLOT

See Figure 12 for a plot of the t-SNE (Van der Maaten & Hinton, 2008) projection for student description embeddings.

E RESULTS ON MORE MODELS

The following table reports mean pass@1 with STUDENTEVAL on 12 models.

Model (Size)	First Failure	Last Failure	First Success	Last Success	HumanEval
Code-Llama-Py-13B (13B)	9.56	9.33	70.22	62.26	42.89
Code-Llama-Py-34B (34B)	11.40	10.14	73.51	64.65	53.29
Code-Llama-Py-7B (7B)	6.51	8.59	66.88	55.36	40.48
GPT-3.5-Turbo-0301 (?)	10.86	12.41	44.84	47.40	48.1
Phi-1 (1.3B)	11.28	8.37	59.16	36.36	51.22
Replit-Code-v1 (2.7B)	3.84	2.83	33.62	18.33	21.09
SantaCoder (1.1B)	2.08	2.11	30.87	21.71	17.81
StarChat-Alpha (15.5B)	10.10	8.78	63.58	51.06	30.03
StarCoderBase (15.5B)	7.82	6.74	65.28	51.74	30.40
StarCoderBase-1B (1B)	1.77	1.21	24.86	13.00	15.17
StarCoderBase-3B (3B)	5.91	5.66	51.73	32.20	21.46
StarCoderBase-7B (7B)	5.49	6.82	62.35	46.42	28.37

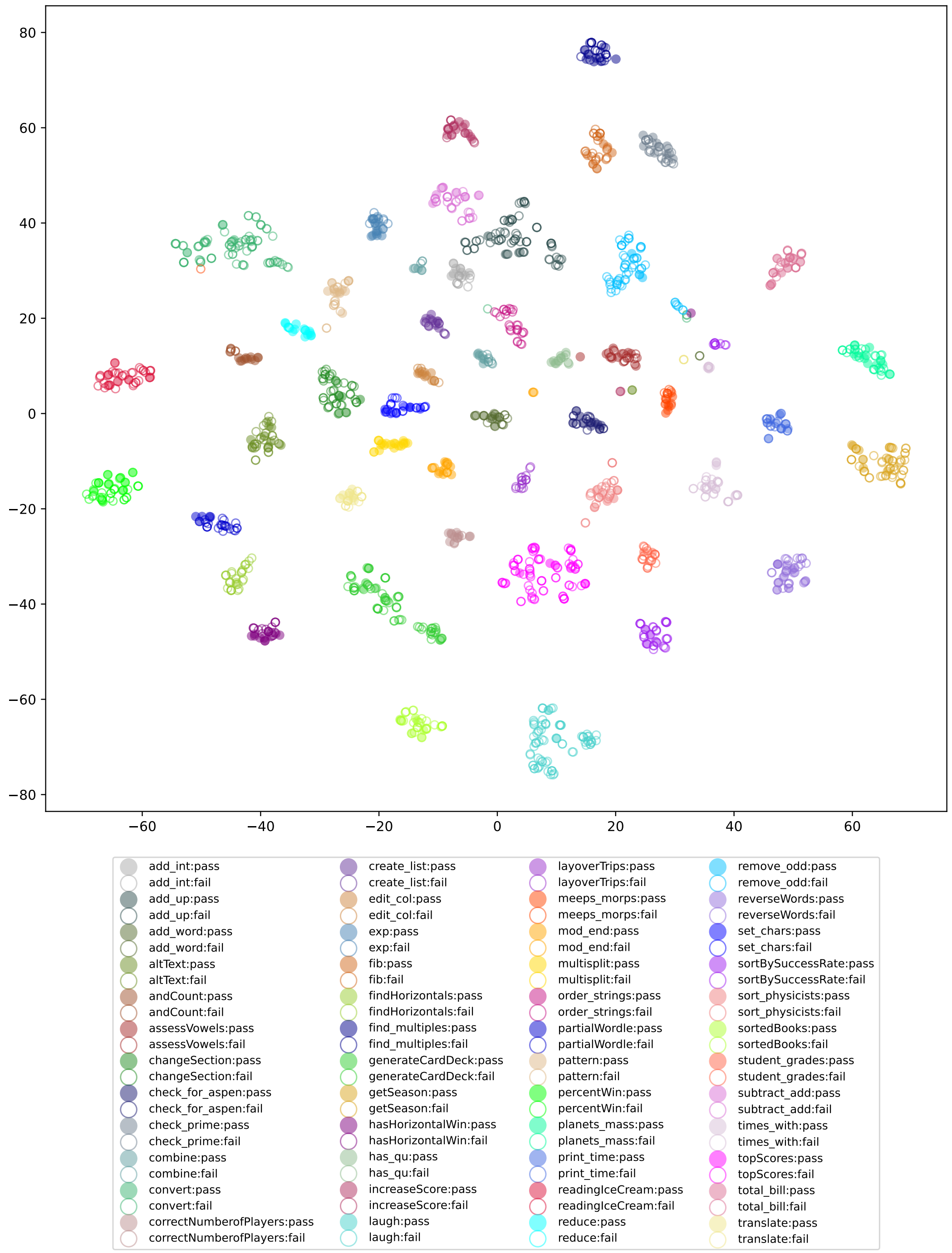


Figure 12: t-SNE projections of student prompt embeddings, colored by problems. Filled and hollow circles represent passed and failed prompts, respectively.