Automatically Repairing Input Data for Novice Python Programs

Madeline Endres, University of Michigan
Why Input-Related Bugs

- Access to 4 years of Python Tutor data thanks to Philip Guo
- 33% of python programs contain a call to `input()`
- Found over 25,000 buggy input / program pairs where only the input differed in the student's "fixed" version
Example Input-Related Error

In practice, some error messages novices face are fixed by only changing the program's input:

Example of Simple Syntactic Mistakes:

Code:

```python
x = float(input())
print(x * math.e / 2)
```

Error Causing Input:

```
5,2
```

Student's Fix:

```
3.1
```

Error = Python expects period decimal notation:

```
ValueError: could not convert string to float: '5,2'
```
More Complex Buggy Input Data Example

Buggy Input:

abcd
*d%#
abacabadaba
#*%*d*%

Error:

```
Traceback (most recent call last):
line 13, in <module>
rashifr_itog += slovar[rashifr[k]]
KeyError: '#'
```
Observations about Input-Related Interpreter Errors

- For syntactic errors, the error message is highly correlated to the eventual student fix.

- For complex errors, fixes are more diverse, but we observed that some fix mutations where more common than others. E.g.:
  - Inserting a string literal from the program
  - Inserting a small integer
  - Swapping two lines of inputs
  - Splitting an input line on whitespace

- Student repairs are generative, not just corrective
  - Often requires multiple error messages to be fixed before finding solution
Research Overview

- Found that a significant fraction of novices programming bugs involve fixing the input data, not just the code itself.
- Developed InFixPy: A tool to automatically repair input bugs in novice Python programs.
- Ran a human study to assess the quality and helpfulness of InFixPy generated repairs.
InFix Algorithm

- Iterative search-based algorithm that modifies the student's error-causing input.
- Use error message templates to try and repair common syntactic errors.
- Apply random additional mutations for non-templated error-messages.
Example of Algorithm Fix

Python Program

```python
def main():
    m=int(input('Inserire un intero '))
    L=list(input('Inserire stringhe '))
    s=''
    s=concatena(m,L)
    print(s)

def concatena(m,L):
    if m==type(int) or L==type(str):
        print('None')
main()
```

Original Bad Input:
ciao

Iteration 1 = ValueError template:
-1

Iteration 2 = Mutation template:
-1
ciao
Human Study Evaluating Repair Quality: Sample Stimulus

Stimulus #6:
The Python program below terminated with an error when run with the shown buggy input. Use the error message and suggested input repair to find the cause of the bug. Then answer the following three questions.

Python Program

```python
monthsdays = {'enero':31, 'febrero':28, 'marzo':31, 'abril':30}
print(monthsdays['febrero'])
print(monthsdays['abril'])
print(monthsdays['Marzo'.lower()])
n = input('Ingrese un mes: ')
print('El mes: ', n.title(), end = ' ')
print('tiene', monthsdays[n.lower()], 'dias')
```

Bug Revealing Input and Error Message

<table>
<thead>
<tr>
<th>Bug Revealing Input</th>
<th>Error Output</th>
</tr>
</thead>
</table>
| 2                   | 28
|                     | 30
|                     | 31
| Ingrese un mes:     | El mes 2
|                     | Traceback [most recent call last]:
|                     | File "temp2018.py", line 7, in <module>
|                     | print ( 'tiene', monthsdays[m.lower()], 'dias' )
|                     | KeyError: '2'

Suggested Input Repair

<table>
<thead>
<tr>
<th>Repaired Input</th>
</tr>
</thead>
<tbody>
<tr>
<td>enero</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Output Produced by Repair</th>
</tr>
</thead>
</table>
| 28
| 30
| 31
| Ingrese un mes: El mes Enero tiene 31 dias |
Evaluation Results

- Empirical results: Can fix **95%** of 25,000 input-related errors

- Human Study results: 97 participants found the machine repairs of *equal helpfulness* and within **4%** the *quality* to student made repairs
Questions?