TABLE I: Root causes and fixing patterns for 15 built-in bug types

Type	ID	Root cause	Example buggy code	Explanation	Fixing patterns
Att	T1-1	Get unhoped type of return variable.	`real_frames. shape[-1].value`	The call to this function would return `int` under eager mode when the shape is a list of ints.	Add an examination to the type.
	T1-2	Infer to an unexisting class member.	Not initialization of class members in the `init` function.	*	Add an examination to the class member.
Imp	T2-1	Not notice API changes.	Code movement, class name changes, or package name changes.	*	Follow the updates in the new package and change the import code, or strict the version of the package to the old one.
Ind	T3-1	Read a file from local.	When reading Keras model in framework *tensorflow*.	*keras* changes it implementation and *tensorflow* haven't adapt to it.	Add a judgment at the reading process to ensure that the layer of the model is correct.
	T3-2	Spilt a string on an empty string.	Split a string, and split again on the substring.	Empty parts would appear after the first splitting, and the second splitting on an empty part would lead to it.	Add length judgment for each part.
Key	T4-1	API misuses.	When reading Keras model in framework *tensorflow*.	*keras* changes it implementation and *tensorflow* haven't adapt to it.	Add a judgment at the reading process to ensure that the layer of the model is correct.
	T4-2	Spilt a string on an empty string.	Split a string, and split again on the substring.	Empty parts would appear after the first splitting, and the second splitting on an empty part would lead to it.	Add length judgment for each part.
Mem	T5-1	Input unique data.	Some specific inputs such as multiple brackets.	During the parsing process to match the brackets, the code would throw it.	Add checks for inputs before compiling, such as checking for the length or illegal characters.
Nam	T6-1	Not find a imported class.	*	Due to updates in third-party packages.	Change the import code.
	T6-2	Reference a variable before assignment.	*	In an enclosing scope, a variable is referenced before the assignment.	Add an initialization code before the variable is referenced.
NoI	T7-1	Input a unsolvable equations.	Caused by unsolvable equations or inequalities in project *sympy*.	*	Add judgments for the inputs, and return another built-in value for the unsolvable inputs.
OSE	T8-1	Read an empty file.	Open an empty file and try to get seek to `byte-1` of the file.	*	Add judgments for empty files and change the range of seeking in a file.
Ove	T9-1	Use floats to calculate.	Too many float numbers are involved in the calculation.	*	Use rational numbers instead of floats.
Rec	T10-1	Recursive call with the same state.	`if a/b==1: call itself`	When `a` and `b` are all 1, the code would always execute the true branch and call itself with the same state.	Add more conditions before the call itself to make the function can finish in the specific inputs.
Run	T11-1	Iterator a dictionary when keys are changed.	`for key in key.items()`	Size changed when iterating a dictionary and the error message is "RuntimeError: dictionary changed size during iteration".	`for key in list(key.items())` in Python 3.x or `for key in key.keys()` in Python 2.x.
Typ	T12-1	Concatenate strings and numbers with a plus.	`a="hello"; b=1; a+b`	Python compiler cannot decide whether to deal with numbers as a string or convert strings to numbers.	Choose to change one type to the same as another.
	T12-2	Call an operation on an object that does not implement it.	*	*	Add code to support the types.
	T12-3	Operate with an unexpected type.	`f1//f2` and `f1/f2`.	An operation is applied to an unexpected type of return value without checking the type.	Add a check for the type of return value before reference it.
	T12-4	Converse type.	The targeted type does not consider the input type.	*	Change the targeted type or implement conversion code for the uncovered type.
Unb	T13-1	Claim a variable only in one block.	`if expr: a=1; else: pass; c=a`	A variable is claimed only in the one branch of a `if` statement.	`a=1; if expr: a=1; else: b=1; c=a`
Val	T14-1	Operate on a variable with an unsupported type.	Operate on the variable and find the inappropriate value.	The operation could support the type, and it would not lead to a *TypeError*.	Passing type information to the operation to help the operation make a judgment before operating, or use type conversion on the arguments.
	T14-2	Receive a negative number.	Function `ptp` in project *numpy* are designed to return an positive value. A negative value is returned under a maximum signed value input.	An operation needs a positive number, but misuse of some transform function would return a negative number.	Fix function to ensure the hoped value is returned.
Zer	T15-1	Divide on float variables	`return f1 / f2`	In some division of two floats, the return value would be zero.	Add a judgment to ensure that `inf` is returned rather than zero.
	T15-2	Not include special case.	*	The code does not cover some special input cases.	Add code to cover special inputs.