Java

```
package Package;
import java.io.BufferedReader;
import java.lang.reflect.Method;
public class Class_01 {
	public static void main(String[] args) {
		Class<?> cla = BufferedReader.class;
		Method[] clamtd = cla.getDeclaredMethods();
		int i = 1;
		for(Method mtd: clamtd) {
			System.out.println(i + ". " + mtd.getName());
			System.out.println("Return Type: " + mtd.getReturnType().getName());
			Class<?>[] parameterTypes = mtd.getParameterTypes();
		System.out.print("Parameter Types: ");
		for (Class<?> paramType : parameterTypes) {
		    System.out.print(paramType.getName() + " ");
		}
		System.out.println("\n");
			i += 1;
		  }
	}
}
```

Output:

```
1. ready
Return Type: boolean
Parameter Types:

2. implReadLine
Return Type: java.lang.String
Parameter Types: boolean [Z

3. implReady
Return Type: boolean
Parameter Types:

4. reset
Return Type: void
Parameter Types:
```

5. lines
Return Type: java.util.stream.Stream
Parameter Types:

6. fill
Return Type: void
Parameter Types:

7. read
Return Type: int
Parameter Types: [C int int

8. read
Return Type: int
Parameter Types:

9. readLine
Return Type: java.lang.String
Parameter Types: boolean [Z

10. readLine
Return Type: java.lang.String
Parameter Types:

11. close
Return Type: void
Parameter Types:

12. mark
Return Type: void
Parameter Types: int

13. skip
Return Type: long
Parameter Types: long

14. markSupported
Return Type: boolean
Parameter Types:

15. implRead
Return Type: int
Parameter Types:

16. implRead
Return Type: int
Parameter Types: [C int int

17. ensureOpen
Return Type: void
Parameter Types:

18. read1
Return Type: int
Parameter Types: [C int int

19. implSkip
Return Type: long
Parameter Types: long

20. implMark
Return Type: void
Parameter Types: int

21. implReset
Return Type: void
Parameter Types:

22. implClose
Return Type: void
Parameter Types:

JavaScript

```
const readline = require('readline');
// Getting the prototype of the readline.Interface class
const readlineMethods = Object.getOwnPropertyNames(readline.Interface.prototype);
// Filtering out non-method properties (e.g., constructor) and printing method names
const methods = readlineMethods.filter(method => typeof readline.Interface.prototype[method]
=== 'function');
console.log('Methods of readline.Interface:');
methods.forEach(method => console.log(method));
```

Output

```
Methods of readline.Interface:
```

constructor
question
_setRawMode
_onLine
_writeToOutput
_addHistory
_refreshLine
_normalWrite
_insertString
_tabComplete
_wordLeft
_wordRight
_deleteLeft
_deleteRight
_deleteWordLeft
_deleteWordRight
_deleteLineLeft
_deleteLineRight
_line
_historyNext
_historyPrev
_getDisplayPos
_getCursorPos
_moveCursor
_ttyWrite

Python

```
# List of all input functions in Python
# input_methods = [input, eval, exec, input().__class__.__call__]
input_methods = [input, eval]
# Loop through each input method, printing its name and documentation
for method in input_methods:
    print(f"Method: {method.__name__}")
    print(f"Documentation: {method.__doc__}")
    print("-" * 50)
```

Output

```
Method: input
Documentation: Read a string from standard input.  The trailing newline is stripped.

The prompt string, if given, is printed to standard output without a
```

trailing newline before reading input.

If the user hits EOF (*nix: Ctrl-D, Windows: Ctrl-Z+Return), raise EOFError.
On *nix systems, readline is used if available.

Method: eval
Documentation: Evaluate the given source in the context of globals and locals.

The source may be a string representing a Python expression
or a code object as returned by compile().
The globals must be a dictionary and locals can be any mapping,
defaulting to the current globals and locals.
If only globals is given, locals defaults to it.
