

main.py



Run

```
1 list = [3, 3, 5, 7, 7, 9, 11, 11]
2 new_list = list(dict.fromkeys(list))
3 print(new_list)
4
```

Automatically Enhancing Error Messages

Christoph Treude, The University of Melbourne

main.py



Run

Shell

Clear

```
1 list = [3, 3, 5, 7, 7, 9, 11, 11]
2 new_list = list(dict.fromkeys(list))
3 print(new_list)
4
```

```
Traceback (most recent call last):
  File "<string>", line 2, in <module>
TypeError: 'list' object is not callable
> |
```

MAXIMS FOR MALFEASANT DESIGNERS, or
HOW TO DESIGN LANGUAGES TO MAKE
PROGRAMMING AS DIFFICULT AS POSSIBLE

Richard L. Wexelblat
Bell Laboratories
Holmdel, NJ 07733

If men could learn from history, what lessons it might teach us!
But passion and party blind our eyes and the light which experience
gives is a lantern on the stern which shines only on the waves behind us.
-Coleridge

KEYWORDS: Programming Language Design
Programming Language Structure
Programming Languages

ABSTRACT

Communication with the computer is by artificial languages: programming languages and command languages, as well as ad hoc languages of messages. While many such languages are sufficiently rich to permit proper expression of what must be said, some are so limited or inconsistent that a user must go to needless effort in learning the language and using it to communicate successfully with the computer.

As part of the final exam of a course on the design of computer languages for human use, students were asked to suggest what "... the language designer can do to make the programming process as difficult as possible."

process and increase the chances of making errors and writing poor programs. The answers tended to stress the particular topics and areas covered in the course, but they represent a reasonable cross-section of the things which the language designer would do well to avoid. Many student responses were duplicates or slight variations on a common theme. After merging similar or closely related responses, there were 29 items, maxims for malfeasant designers. For convenience, I have grouped them into five categories:

- Program writing and formatting
- Program and control structures

Maxims for malfeasant designers, or how to design languages to make programming as difficult as possible

Richard L. Wexelblat

If men could learn from history, what lessons it might teach us!
But passion and party blind our eyes and the light which experience
gives is a lantern on the stern which shines only on the waves behind us.
-Coleridge

KEYWORDS: Programming Language Design
Programming Language Structure
Programming Languages

ABSTRACT

Communication with the computer is by artificial languages: programming languages and command languages, as well as ad hoc languages of messages. While many such languages are sufficiently rich to permit proper expression of what must be said, some are so limited or inconsistent that a user must go to needless effort in learning the language and using it to communicate successfully with the computer.

As part of the final exam of a course on the design of computer languages for human use, students were asked to suggest what "... the language designer can do to make the programming process as difficult as possible."

process and increase the chances of making errors and writing poor programs. The answers tended to stress the particular topics and areas covered in the course, but they represent a reasonable cross-section of the things which the language designer would do well to avoid. Many student responses were duplicates or slight variations on a common theme. After merging similar or closely related responses, there were 29 items, maxims for malfeasant designers. For convenience, I have grouped them into five categories:

- Program writing and formatting
- Program and control structures

Maxims for malfeasant designers, or how to design languages to make programming as difficult as possible

Richard L. Wexelblat

Use cryptic diagnostics

To maximize difficulty for the user, it is important that the diagnostic messages reflect what the program did, rather than what the user did.

main.py



Run

Shell

Clear

```
1 list = [3, 3, 5, 7, 7, 9, 11, 11]
2 new_list = list(dict.fromkeys(list))
3 print(new_list)
4
```

```
Traceback (most recent call last):
  File "<string>", line 2, in <module>
TypeError: 'list' object is not callable
> |
```

main.py



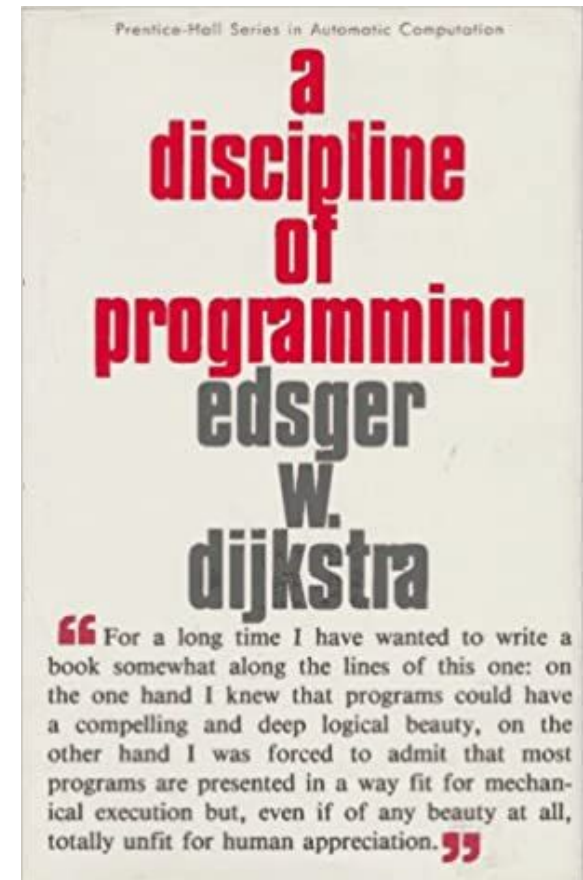
Run

Shell

Clear

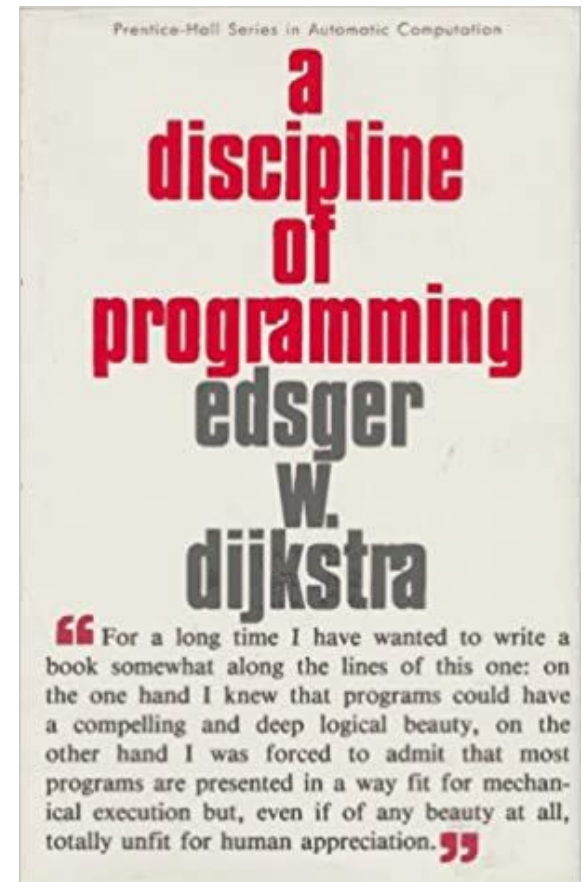
```
1 list = [3, 3, 5, 7, 7, 9, 11, 11]
2 new_list = list(dict.fromkeys(list))
3 print(new_list)
4
```

```
Traceback (most recent call last):
  File "<string>", line 2, in <module>
TypeError: 'list' object is not callable
> |
```



```
main.py ☐ ☾ Run Shell Clear  
1 list = [3, 3, 5, 7, 7, 9, 11, 11]  
2 new_list = list(dict.fromkeys(list))  
3 print(new_list)  
4
```

Traceback (most recent call last):
File "<string>", line 2, in <module>
TypeError: 'list' object is not callable
> |




```
main.py ⌂ ☾ Run Shell Clear  
1 list = [3, 3, 5, 7, 7, 9, 11, 11]  
2 new_list = list(dict.fromkeys(list))  
3 print(new_list)  
4
```

Traceback (most recent call last):
File "<string>", line 2, in <module>
TypeError: 'list' object is not callable
> |

▲ It should work fine. Don't use `tuple`, `list` or other special names as a variable name. It's probably what's causing your problem.

939



```
>>> l = [4,5,6]  
>>> tuple(l)  
(4, 5, 6)  
  
>>> tuple = 'whoops' # Don't do this  
>>> tuple(l)  
TypeError: 'tuple' object is not callable
```



stackoverflow

main.py



Run



Shell

Clear

```
1 list = [3, 3, 5, 7, 7, 9, 11, 11]
2 new_list = list(dict.fromkeys(list))
3 print(new_list)
4
```

```
Traceback (most recent call last):
  File "<string>", line 2, in <module>
TypeError: 'list' object is not callable
> |
```



main.py			Run	Shell	<input type="button" value="Clear"/>
<pre>1 list = [3, 3, 5, 7, 7, 9, 11, 11] 2 new_list = list(dict.fromkeys(list)) 3 print(new_list) 4</pre>				<pre>Traceback (most recent call last): File "<string>", line 2, in <module> TypeError: 'list' object is not callable > </pre>	

Error Message Parsing



```
main.py ⌵ 🌙 Run Shell Clear  
1 list = [3, 3, 5, 7, 7, 9, 11, 11]  
2 new_list = list(dict.fromkeys(list))  
3 print(new_list)  
4
```

Traceback (most recent call last):
File "<string>", line 2, in <module>
TypeError: 'list' object is not callable
> |

Error Message Parsing

Query Construction



```
main.py [Full Screen] [Dark Mode] [Run] Shell [Clear]
1 list = [3, 3, 5, 7, 7, 9, 11, 11]
2 new_list = list(dict.fromkeys(list))
3 print(new_list)
4
```

Traceback (most recent call last):
File "<string>", line 2, in <module>
TypeError: 'list' object is not callable
> |

Error Message Parsing

Query Construction

Answer Selection



```
main.py ⌵ 🌙 Run Shell Clear  
1 list = [3, 3, 5, 7, 7, 9, 11, 11]  
2 new_list = list(dict.fromkeys(list))  
3 print(new_list)  
4
```

Traceback (most recent call last):
 File "<string>", line 2, in <module>
TypeError: 'list' object is not callable
> |

Error Message Parsing

Query Construction

Answer Selection

Customisation



```
main.py ⌂ 🌙 Run Shell Clear  
1 list = [3, 3, 5, 7, 7, 9, 11, 11]  
2 new_list = list(dict.fromkeys(list))  
3 print(new_list)  
4
```

Traceback (most recent call last):
File "<string>", line 2, in <module>
TypeError: 'list' object is not callable
> |

Error Message Parsing

Query Construction

Answer Selection

Customisation

Summarisation



main.py



Run

```
1 list = [3, 3, 5, 7, 7, 9, 11, 11]
2 new_list = list(dict.fromkeys(list))
3 print(new_list)
4
```

Error Message Parsing

Query Construction

Answer Selection

Customisation

Summarisation



Shell

Clear

```
Don't use tuple, list or other
special names as a variable name.
It's probably what's causing your
problem.
```


main.py



Run

```
1 list = [3, 3, 5, 7, 7, 9, 11, 11]
2 new_list = list(dict.fromkeys(list))
3 print(new_list)
4
```

Error Message Parsing

Query Construction

Answer Selection

Customisation

Summarisation



Pycee:

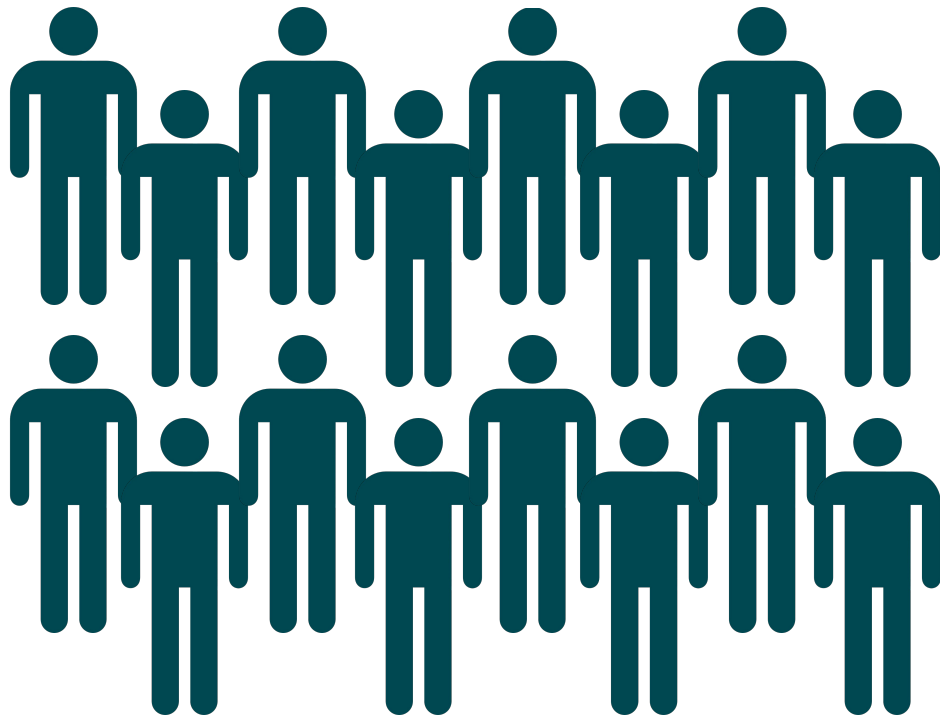
Shell

Clear

```
Don't use tuple, list or other
special names as a variable name.
It's probably what's causing your
problem.
```

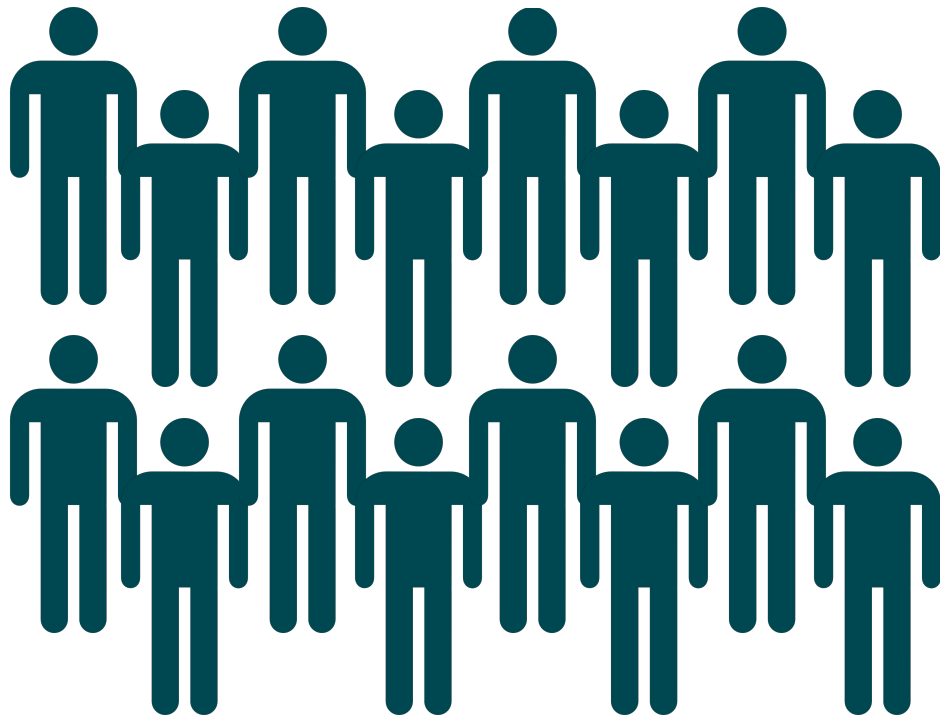
How do programmers perceive Pycee?





All Exercises

- 1: [Character Input](#) 🍌
- 2: [Odd Or Even](#) 🍌
- 3: [List Less Than Ten](#) 🍌🍌
- 4: [Divisors](#) 🍌🍌
- 5: [List Overlap](#) 🍌🍌
- 6: [String Lists](#) 🍌🍌
- 7: [List Comprehensions](#) 🍌🍌
- 8: [Rock Paper Scissors](#) 🍌🍌🍌
- 9: [Guessing Game One](#) 🍌🍌🍌
- 10: [List Overlap Comprehensions](#) 🍌🍌
- 11: [Check Primality Functions](#) 🍌🍌🍌
- 12: [List Ends](#) 🍌
- 13: [Fibonacci](#) 🍌🍌
- 14: [List Remove Duplicates](#) 🍌🍌
- 15: [Reverse Word Order](#) 🍌🍌🍌
- 16: [Password Generator](#) 🍌🍌🍌🍌
- 17: [Decode A Web Page](#) 🍌🍌🍌🍌
- 18: [Cows And Bulls](#) 🍌🍌🍌
- 19: [Decode A Web Page Two](#) 🍌🍌🍌🍌
- 20: [Element Search](#) 🍌
- 21: [Write To A File](#) 🍌
- 22: [Read From File](#) 🍌



All Exercises

- 1: [Character Input](#) 🐍
- 2: [Odd Or Even](#) 🐍
- 3: [List Less Than Ten](#) 🐍🐍
- 4: [Divisors](#) 🐍🐍
- 5: [List Overlap](#) 🐍🐍
- 6: [String Lists](#) 🐍🐍
- 7: [List Comprehensions](#) 🐍🐍
- 8: [Rock Paper Scissors](#) 🐍🐍🐍
- 9: [Guessing Game One](#) 🐍🐍🐍
- 10: [List Overlap Comprehensions](#) 🐍🐍

Table of Contents

Built-in Exceptions

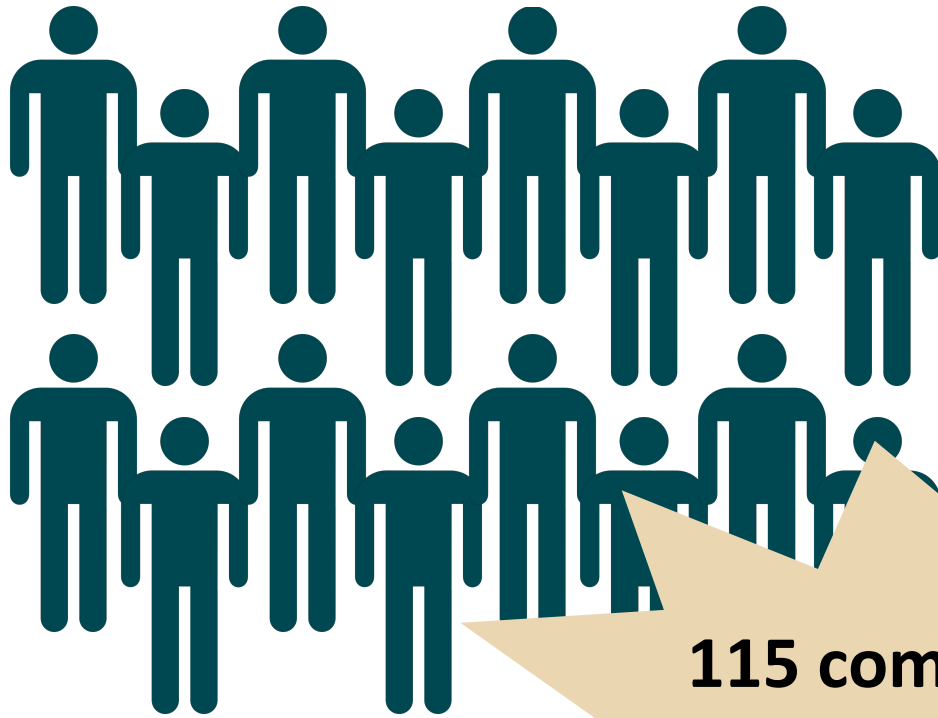
- Exception context
- Inheriting from built-in exceptions
- Base classes
- Concrete exceptions
 - OS exceptions
- Warnings
- Exception hierarchy

Previous topic

Built-in Exceptions

In Python, all exceptions must be instances of a class that derives from `BaseException`. In a `try` statement with an `except` clause that mentions a particular class, that clause also handles any exception classes derived from that class (but not exception classes from which *it* is derived). Two exception classes that are not related via subclassing are never equivalent, even if they have the same name.

The built-in exceptions listed below can be generated by the interpreter or built-in functions. Except where mentioned, they have an “associated value” indicating the detailed cause of the error. This may be a string or a tuple of several items of information (e.g., an error code and a string explaining the code). The associated value is usually passed as arguments to the exception class’s constructor.



115 compiler errors

All Exercises

- 1: [Character Input](#) 🐍
- 2: [Odd Or Even](#) 🐍
- 3: [List Less Than Ten](#) 🐍🐍
- 4: [Divisors](#) 🐍🐍
- 5: [List Overlap](#) 🐍🐍
- 6: [Long Lists](#) 🐍🐍
- 7: [List Comprehensions](#) 🐍🐍
- 8: [Rock Paper Scissors](#) 🐍🐍🐍
- 9: [Guessing Game One](#) 🐍🐍🐍
- 10: [List Overlap Comprehensions](#) 🐍🐍

Table of Contents

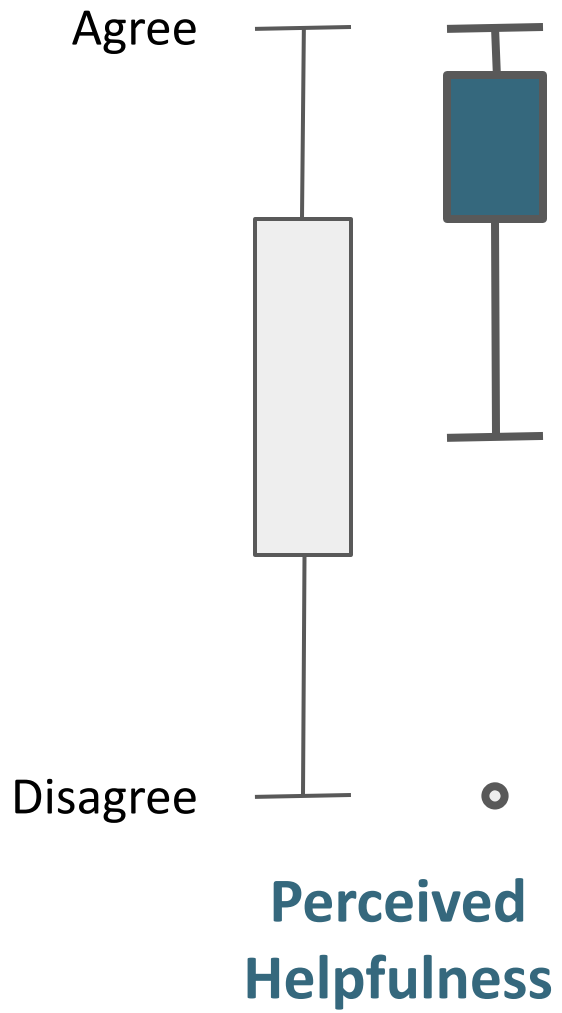
- Built-in Exceptions
 - Exception context
 - Inheriting from built-in exceptions
 - Base classes
 - Concrete exceptions
 - OS exceptions
 - Warnings
 - Exception hierarchy

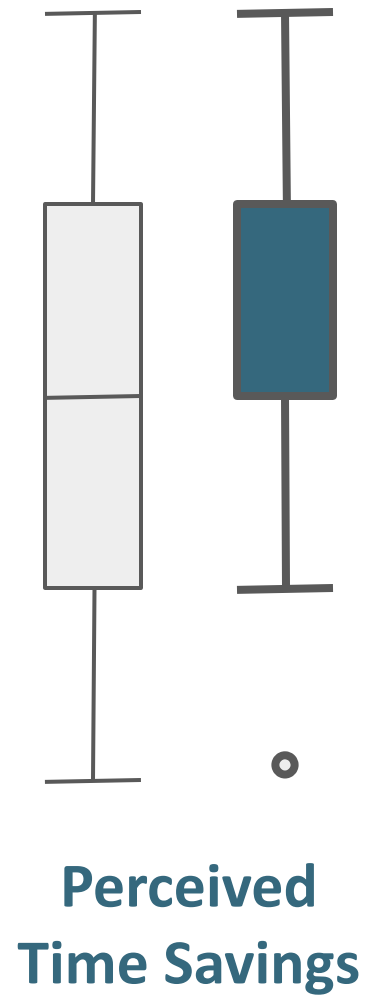
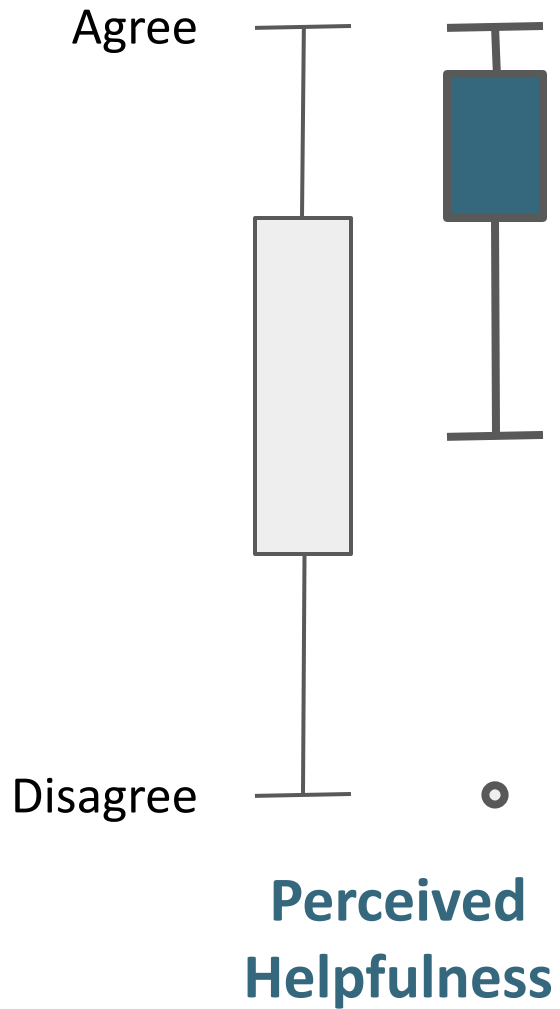
Previous topic

Built-in Exceptions

In Python, all exceptions must be instances of a class that derives from `BaseException`. In a `try` statement with an `except` clause that mentions a particular class, that clause also handles any exception classes derived from that class (but not exception classes from which *it* is derived). Two exception classes that are not related via subclassing are never equivalent, even if they have the same name.

The built-in exceptions listed below can be generated by the interpreter or built-in functions. Except where mentioned, they have an “associated value” indicating the detailed cause of the error. This may be a string or a tuple of several items of information (e.g., an error code and a string explaining the code). The associated value is usually passed as arguments to the exception class’s constructor.





main.py





Run

Shell

Clear

```
1 list = [3, 3, 5, 7, 7, 9, 11, 11]
2 new_list = list(dict.fromkeys(list))
3 print(new_list)
4
```

Don't use tuple, list or other special names as a variable name. It's probably what's causing your problem.

main.py			Run	Shell	Clear
<pre>1 list = [3, 3, 5, 7, 7, 9, 11, 11] 2 new_list = list(dict.fromkeys(list)) 3 print(new_list) 4</pre>				<p>Don't use tuple, list or other special names as a variable name. It's probably what's causing your problem.</p>	

Crowdsource your error messages
and we'll develop tools to automate the integration