Curtin University

Input Output

ISYS2001, School of Marketing and Management

Curtin University

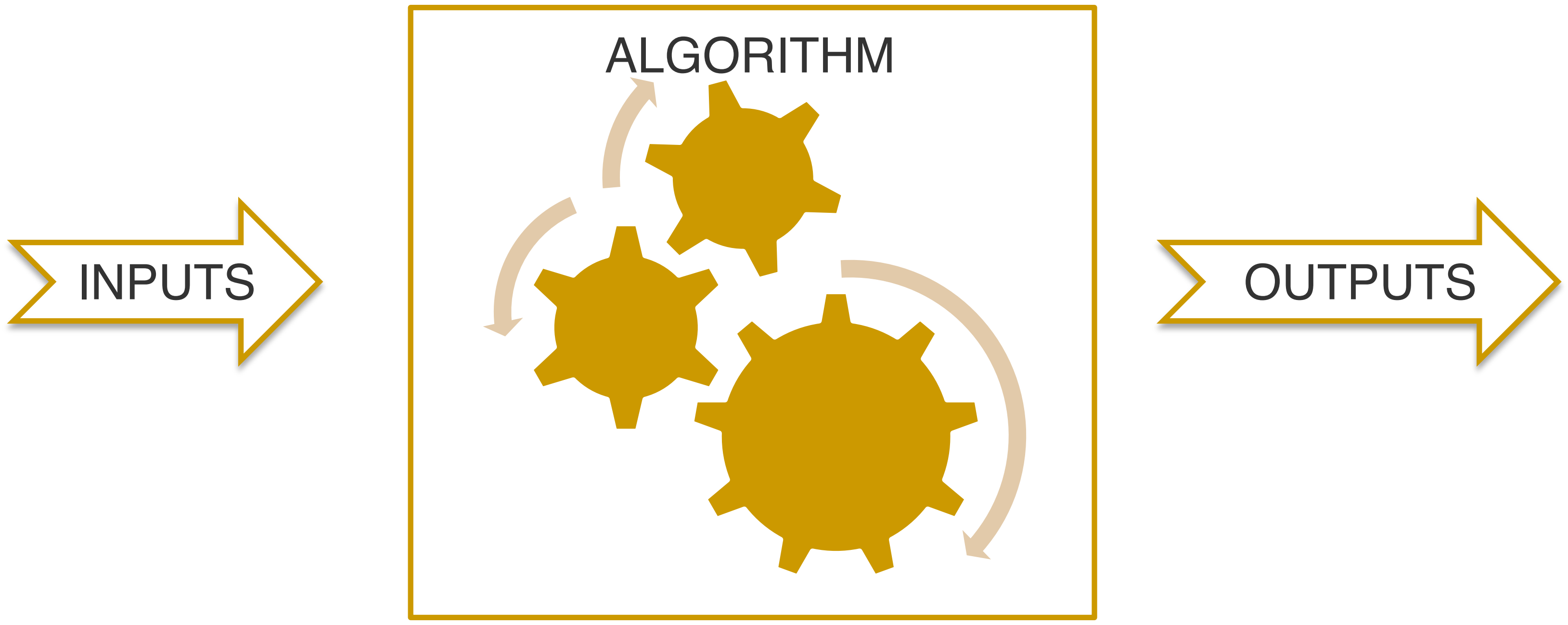
I acknowledge the traditional custodians of the land on which I work and live, and recognise their continuing connection to land, water and community. I pay respect to elders past, present and emerging.

Curtin University

Today

- Getting Help
- Handling Exceptions
- Validating Input
- File Input/Output

Input Output Model



Six Things

- Input
- Output
- Calculate (Add, multiply, less than etc.)
- Store (assignment)
- Decide (if-then)
- Repeat (for, while)

Core Principles (so far)

- Algorithms
  - Expressed in structured English (Pseudocode)
  - Sequence (must put things in correct order)
  - Selection (if-then-else)
  - Repetition (while)
- Encapsulation (group things)
  - Manage complexity
  - Data example – lists group values/variables
  - Code example – function group expressions

Python

- Values (Literals)
- Data Types (everything has a datatype)
    - str (string)
    - int (integer)
    - float (real number, decimal number)
    - list []
- Expressions
    - Mathematical operations → evaluates to value
    - Relational Operators → Boolean Expressions
    - Assignment. Store/save a value to a variable
    - if-else (selection)
    - Repetition (for)
- Functions
    - builtins
    - import (other packages to extend python)

(Our) Development Environment

- Python Notebook
  - Code Cells
  - Text Cells (Markdown)
  - Interactively run cells
  - Output in notebook
- Workflow
  - Edit in notebook
  - Frequently save to GitHub

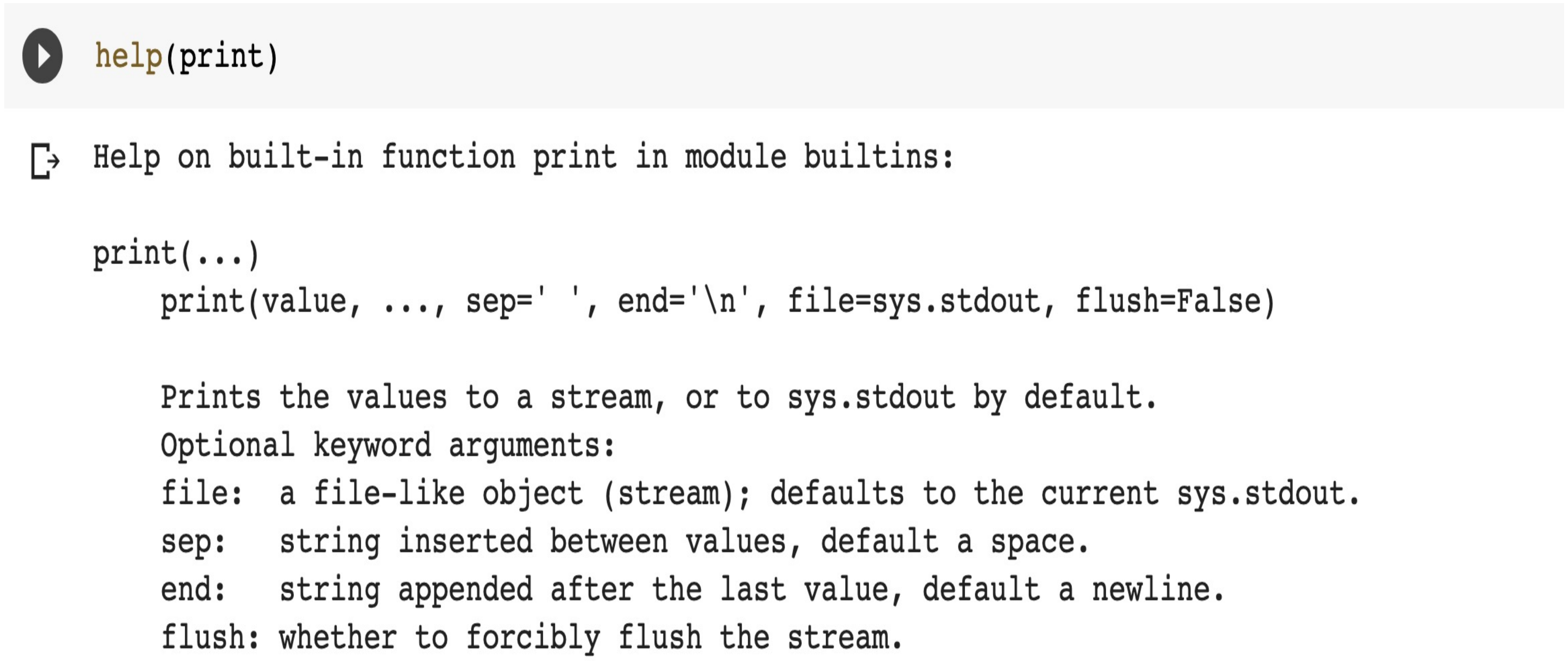
Getting Help

- dir([module]) – list function available
- help([object]) – list documentation
- ? And ??
- Help others (and your future-self) by using doc string
- Online

Note: print() and type() are helpful but relate to the concept of testing. We have been 'testing' but will formalise this in later module.

```python
help()
```

```
Welcome to Python 3.7's help utility!

If this is your first time using Python, you should definitely check out
the tutorial on the Internet at https://docs.python.org/3.7/tutorial/.

Enter the name of any module, keyword, or topic to get help on writing
Python programs and using Python modules.  To quit this help utility and
return to the interpreter, just type "quit".

To get a list of available modules, keywords, symbols, or topics, type
"modules", "keywords", "symbols", or "topics".  Each module also comes
with a one-line summary of what it does; to list the modules whose name
or summary contain a given string such as "spam", type "modules spam".

help>
```

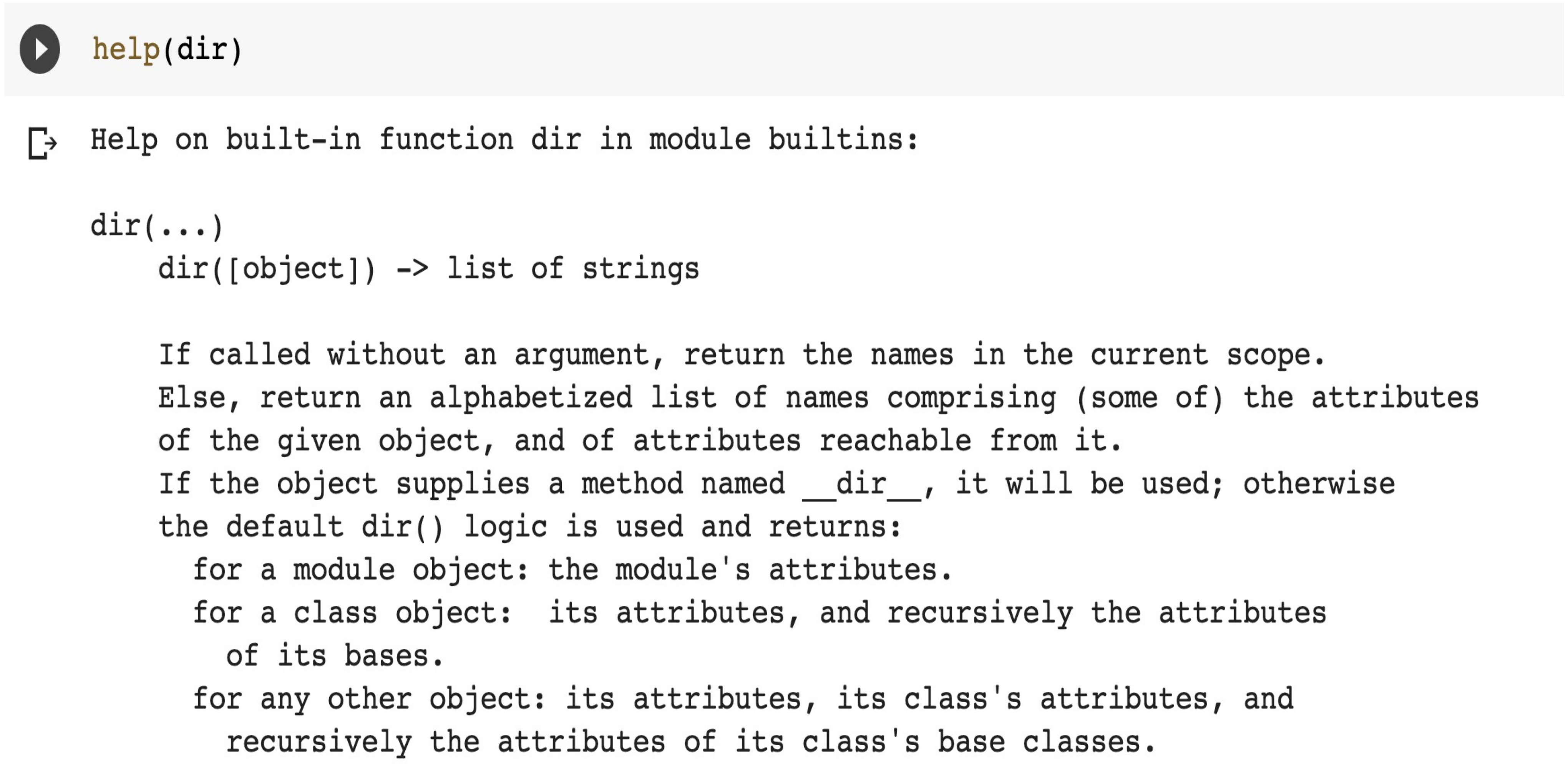


```python
help(print)
```

```
Help on built-in function print in module builtins:

print(...)
    print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)

    Prints the values to a stream, or to sys.stdout by default.
    Optional keyword arguments:
    file:  a file-like object (stream); defaults to the current sys.stdout.
    sep:   string inserted between values, default a space.
    end:   string appended after the last value, default a newline.
    flush: whether to forcibly flush the stream.
```

```python
import io
dir(io)
```

```
['BlockingIOError',
 'BufferedIOBase',
 'BufferedRWPair',
 'BufferedRandom',
 'BufferedReader',
 'BufferedWriter',
 'BytesIO',
 'DEFAULT_BUFFER_SIZE',
 'FileIO',
 'IOBase',
 'IncrementalNewlineDecoder',
 'OpenWrapper',
 'RawIOBase',
 'SEEK_CUR',
 'SEEK_END',
 'SEEK_SET',
 'StringIO',
```

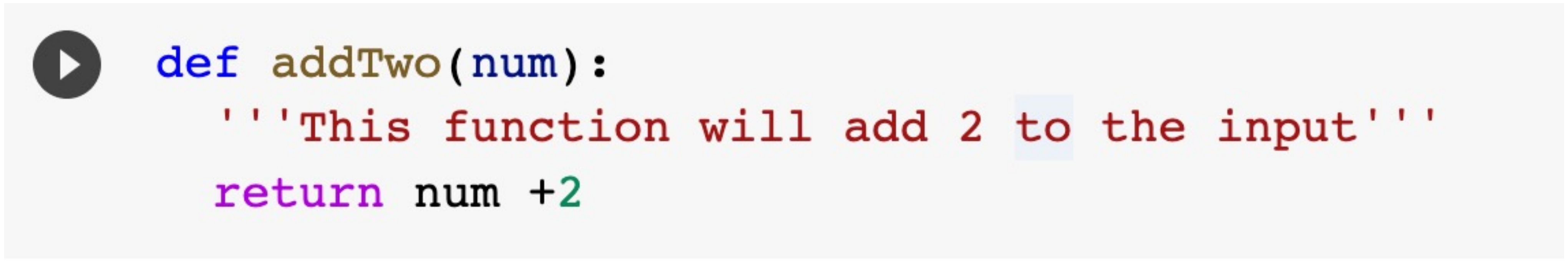


```python
help(dir)
```

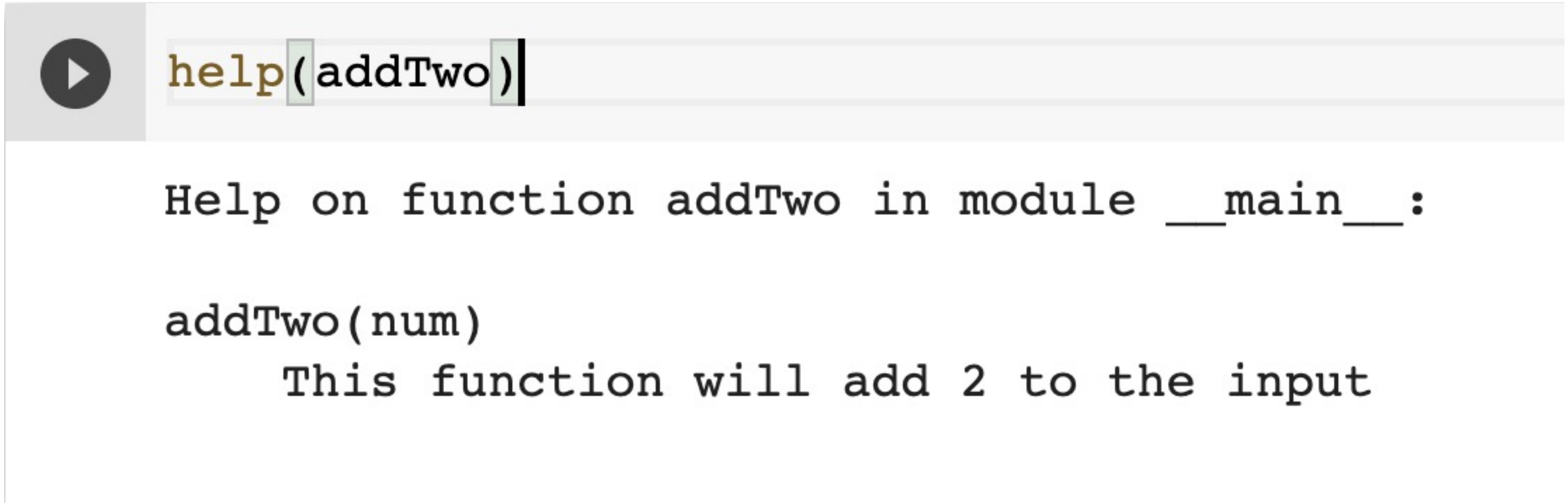
```
Help on built-in function dir in module builtins:

dir(...)
    dir([object]) -> list of strings

    If called without an argument, return the names in the current scope.
    Else, return an alphabetized list of names comprising (some of) the attributes
    of the given object, and of attributes reachable from it.
    If the object supplies a method named __dir__, it will be used; otherwise
    the default dir() logic is used and returns:
      for a module object: the module's attributes.
      for a class object:  its attributes, and recursively the attributes
        of its bases.
      for any other object: its attributes, its class's attributes, and
        recursively the attributes of its class's base classes.
```

```
help(io.open)
```

```
Help on built-in function open in module io:

open(file, mode='r', buffering=-1, encoding=None, errors=None, newline=None, closefd=True, opener=None)
    Open file and return a stream.  Raise OSError upon failure.

    file is either a text or byte string giving the name (and the path
    if the file isn't in the current working directory) of the file to
    be opened or an integer file descriptor of the file to be
    wrapped. (If a file descriptor is given, it is closed when the
    returned I/O object is closed, unless closefd is set to False.)

    mode is an optional string that specifies the mode in which the file
    is opened. It defaults to 'r' which means open for reading in text
    mode.  Other common values are 'w' for writing (truncating the file if
    it already exists), 'x' for creating and writing to a new file, and
    'a' for appending (which on some Unix systems, means that all writes
    append to the end of the file regardless of the current seek position).
    In text mode, if encoding is not specified the encoding used is platform
    dependent: locale.getpreferredencoding(False) is called to get the
    current locale encoding. (For reading and writing raw bytes use binary
    mode and leave encoding unspecified.) The available modes are:
```



```python
def addTwo(num):
  '''This function will add 2 to the input'''
  return num +2
```



```python
help(addTwo)
```

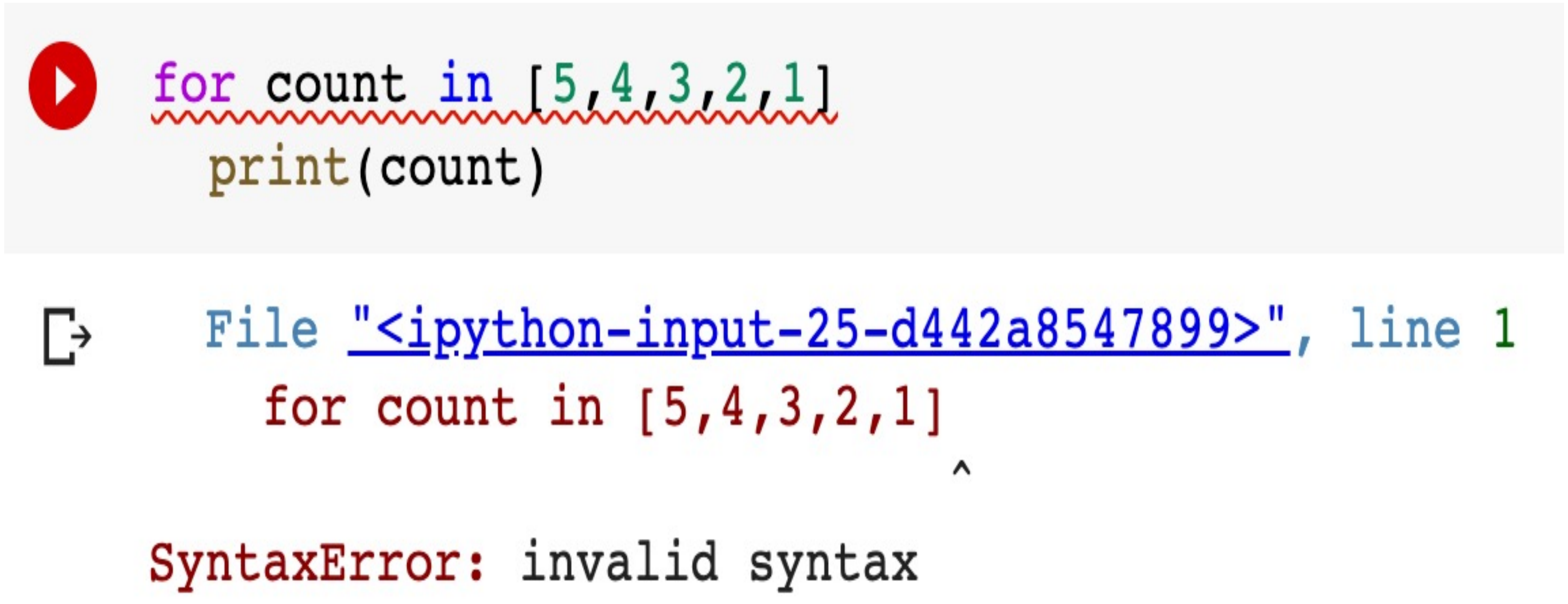
```
Help on function addTwo in module __main__:

addTwo(num)
    This function will add 2 to the input
```

Getting Help

- Internet Search
- dir([module]) – list function available
- help([object]) – list documentation
- Add doc string when creating function

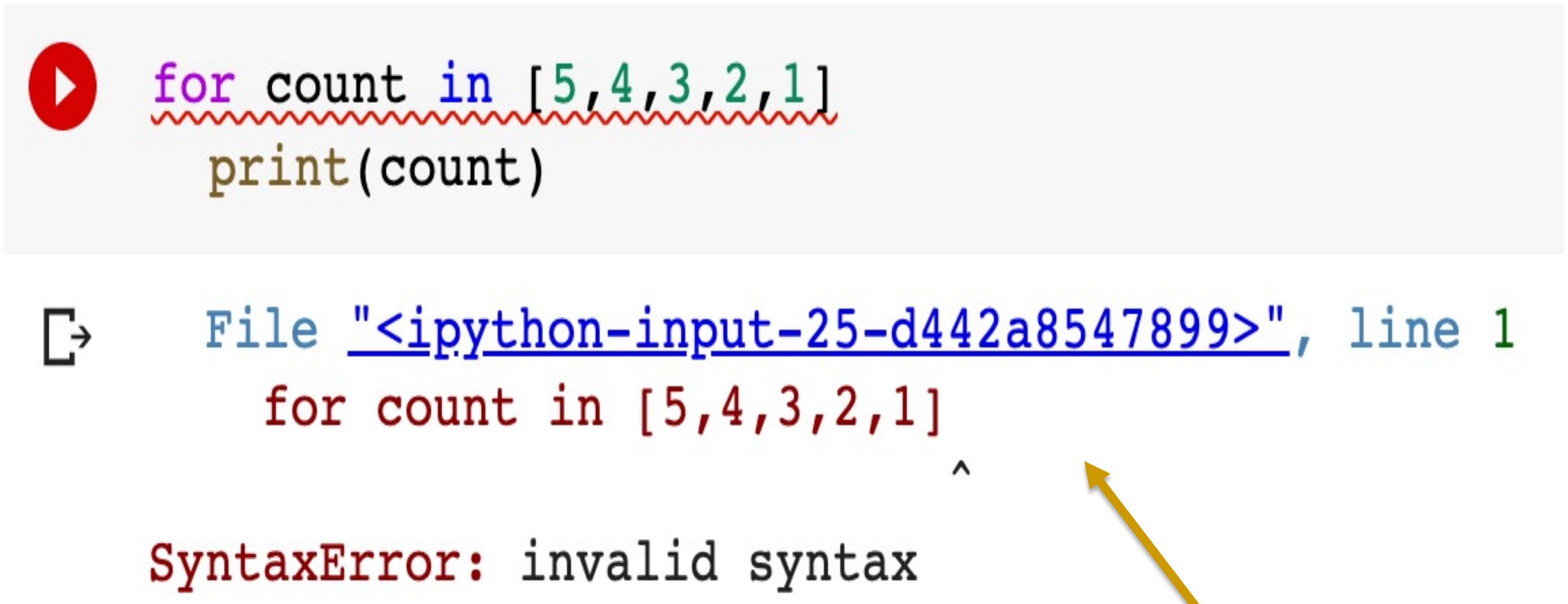
Exception

Exception Object:
- Description
- Traceback



```python
for count in [5,4,3,2,1]
  print(count)
```

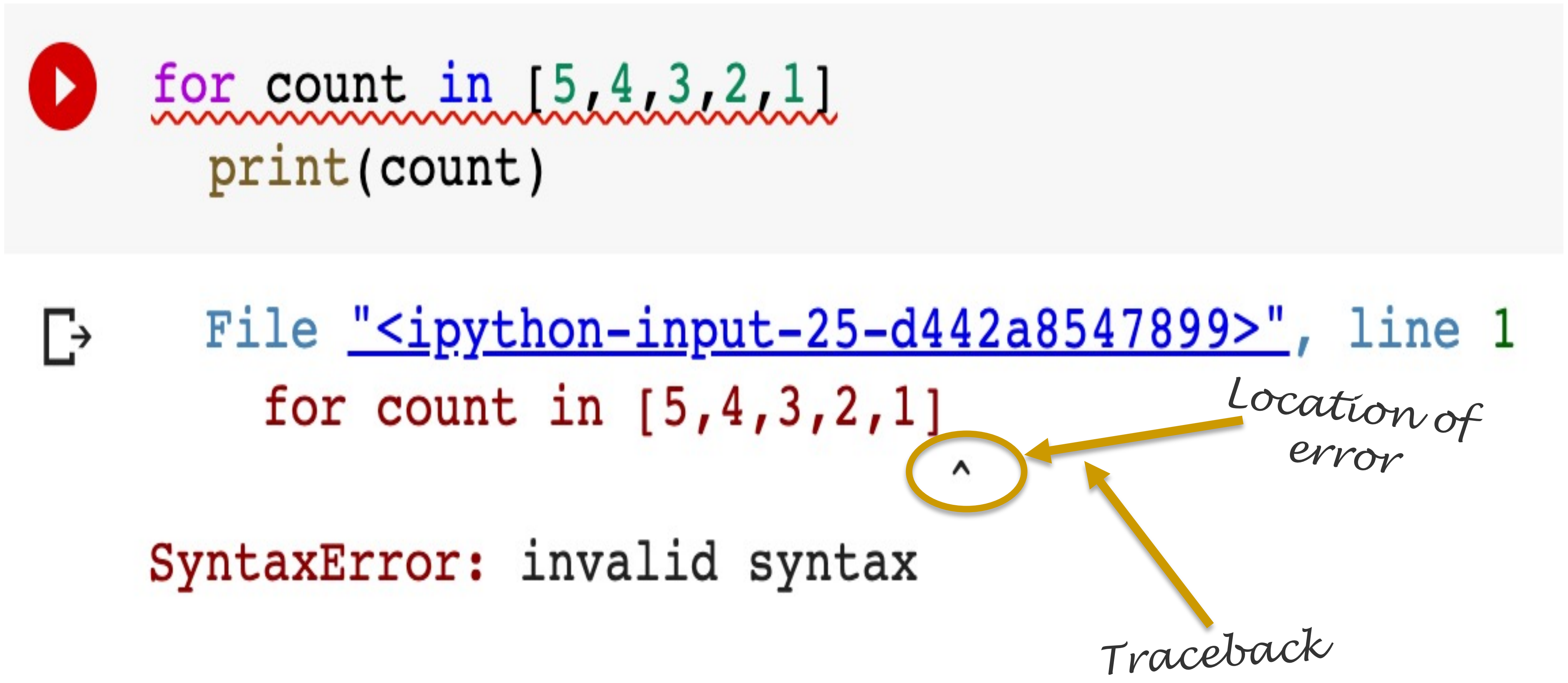
```
  File "<ipython-input-25-d442a8547899>", line 1
    for count in [5,4,3,2,1]
                            ^
SyntaxError: invalid syntax
```



```
for count in [5,4,3,2,1]
  print(count)
```

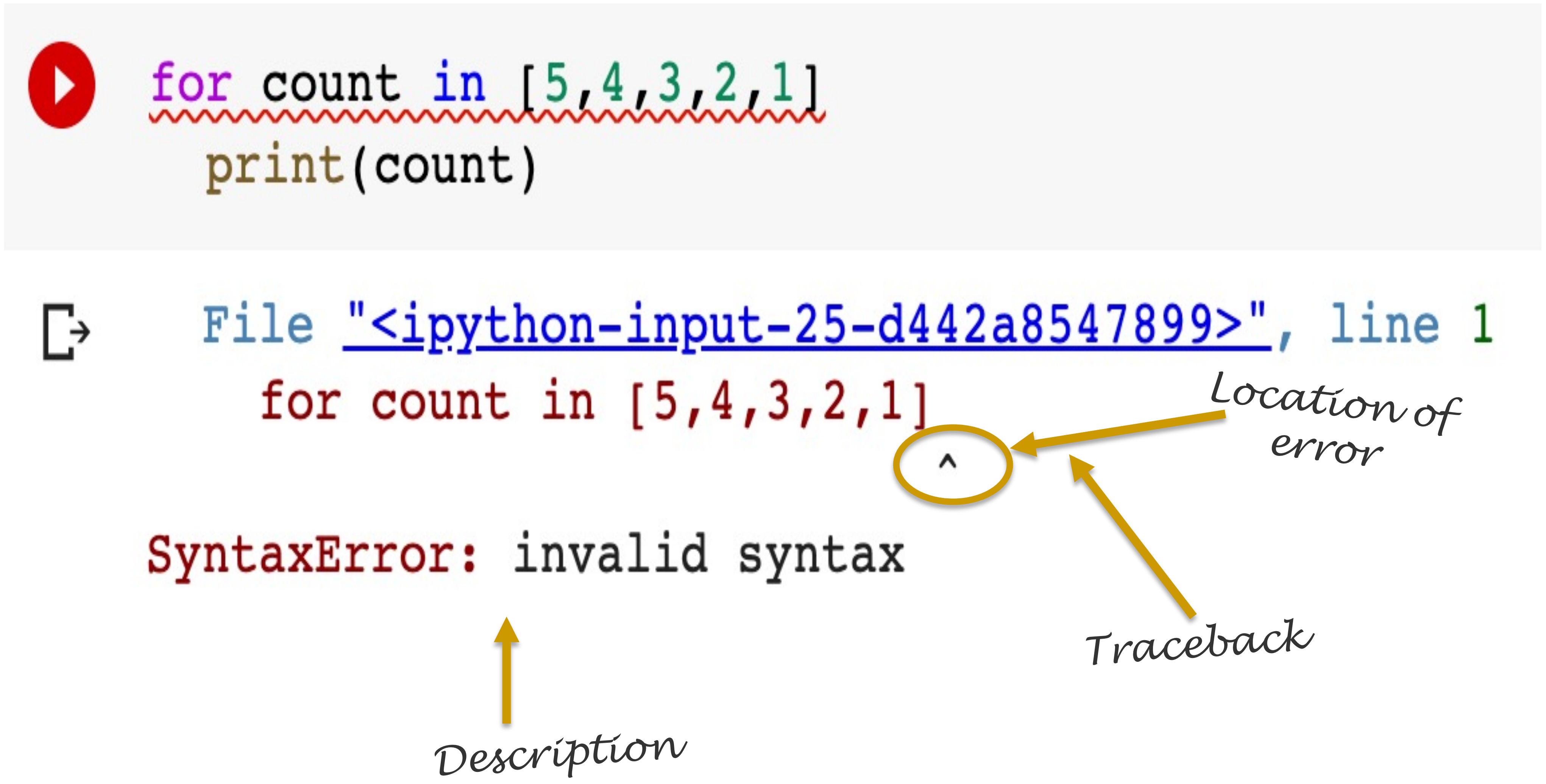
```
  File "<ipython-input-25-d442a8547899>", line 1
    for count in [5,4,3,2,1]
                            ^
SyntaxError: invalid syntax
```

Traceback



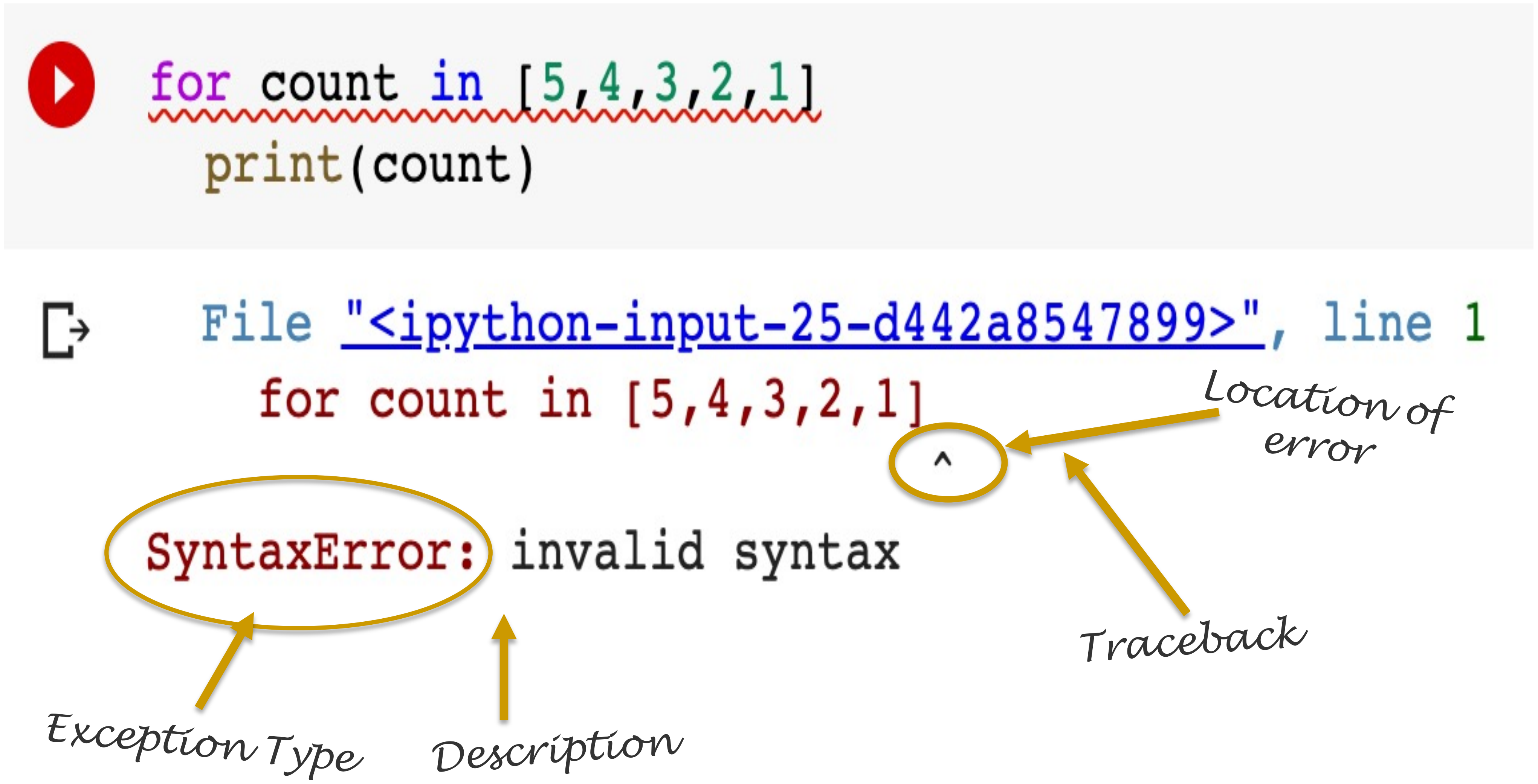
```
for count in [5,4,3,2,1]
   print(count)
```

```
  File "<ipython-input-25-d442a8547899>", line 1
    for count in [5,4,3,2,1]
                            ^

SyntaxError: invalid syntax
```



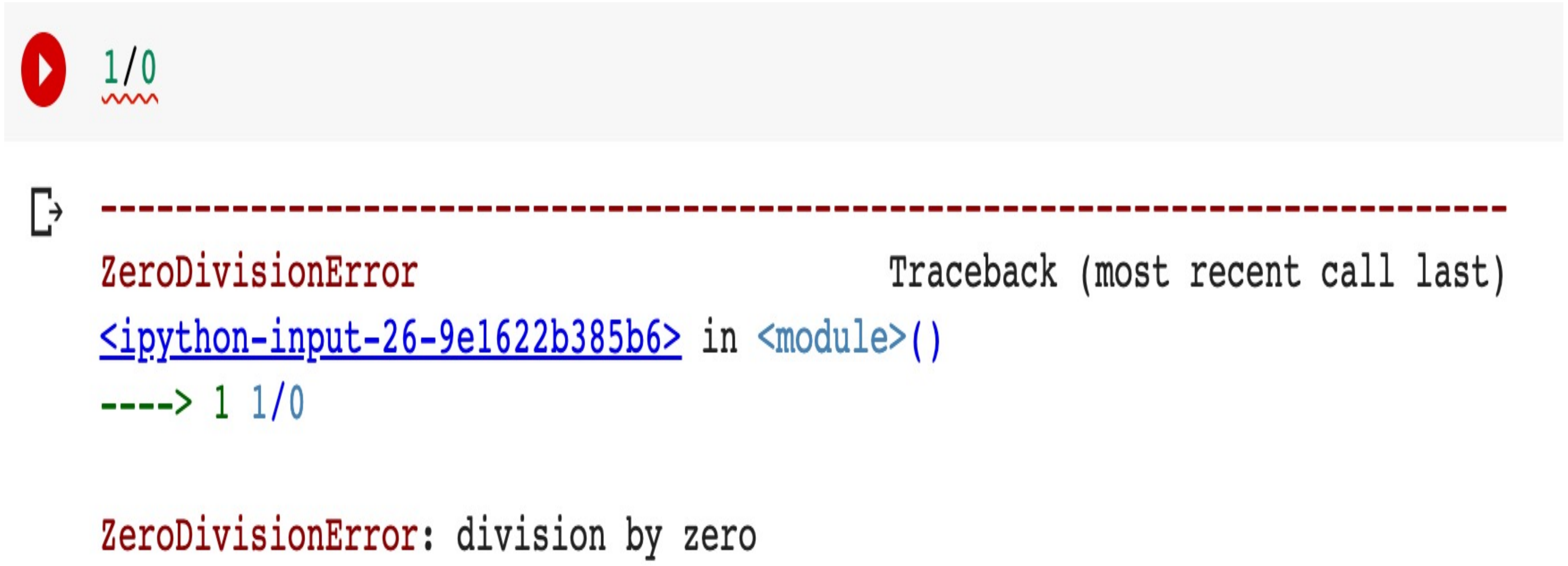
```
for count in [5,4,3,2,1]
  print(count)
```

```
  File "<ipython-input-25-d442a8547899>", line 1
    for count in [5,4,3,2,1]
                            ^
SyntaxError: invalid syntax
```



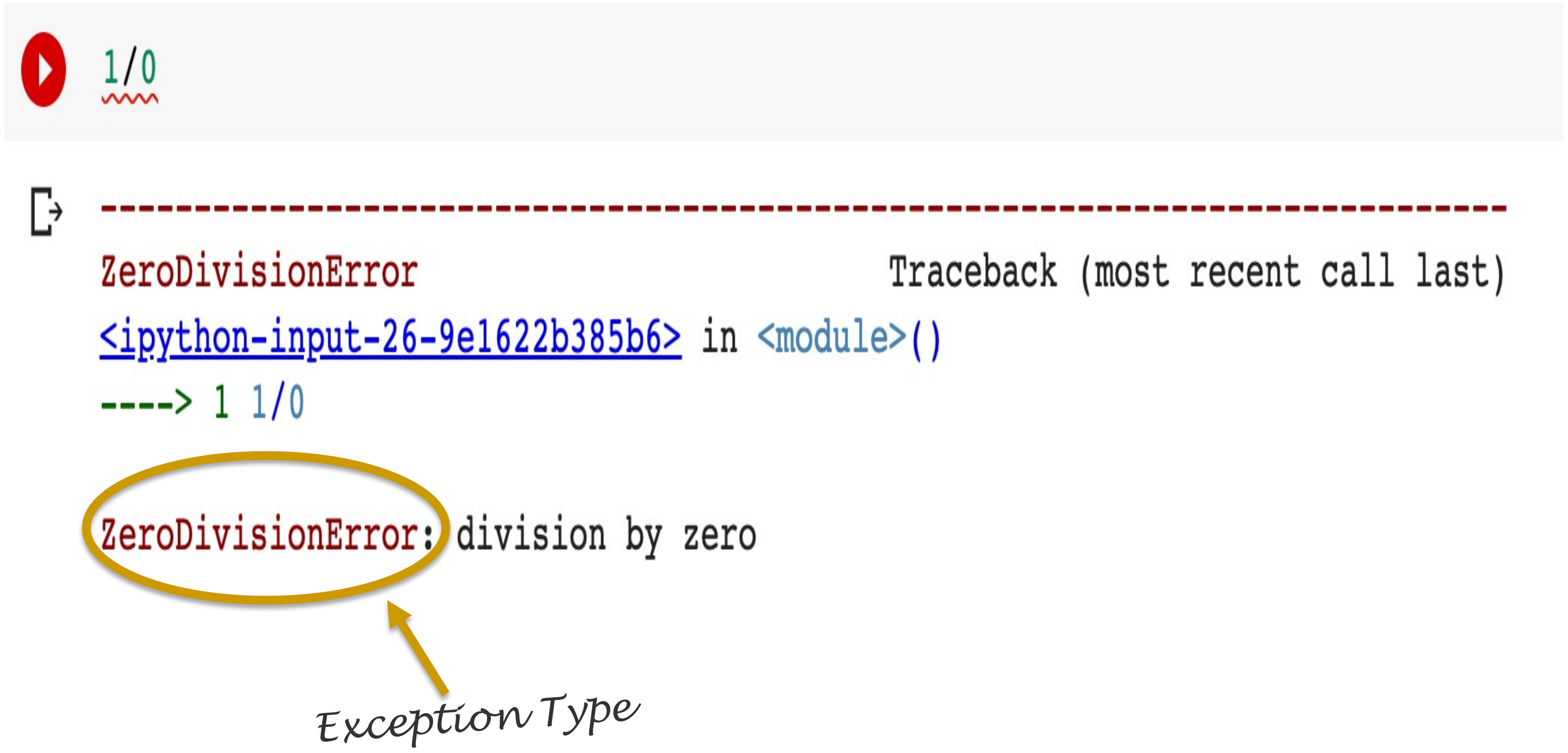
```
for count in [5,4,3,2,1]
    print(count)
```

```
  File "<ipython-input-25-d442a8547899>", line 1
    for count in [5,4,3,2,1]
                            ^
SyntaxError: invalid syntax
```



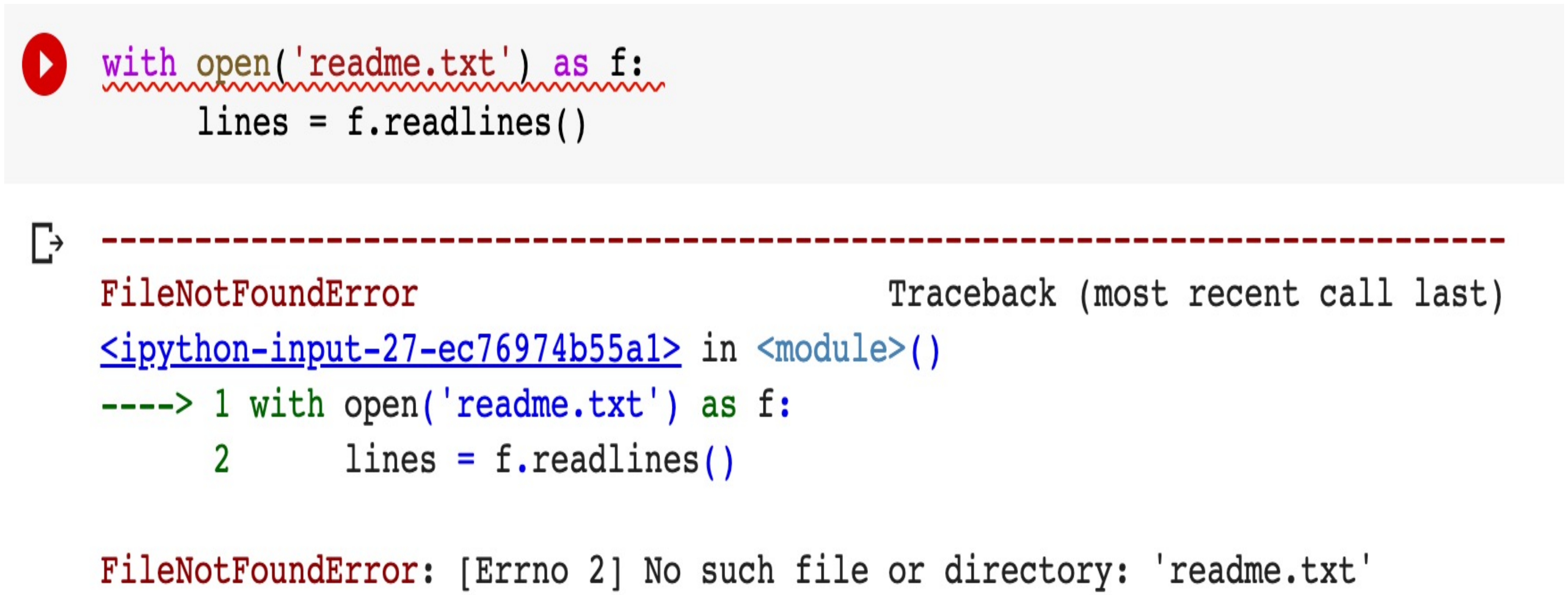
```
1/0
```

```
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
<ipython-input-26-9e1622b385b6> in <module>()
----> 1 1/0

ZeroDivisionError: division by zero
```



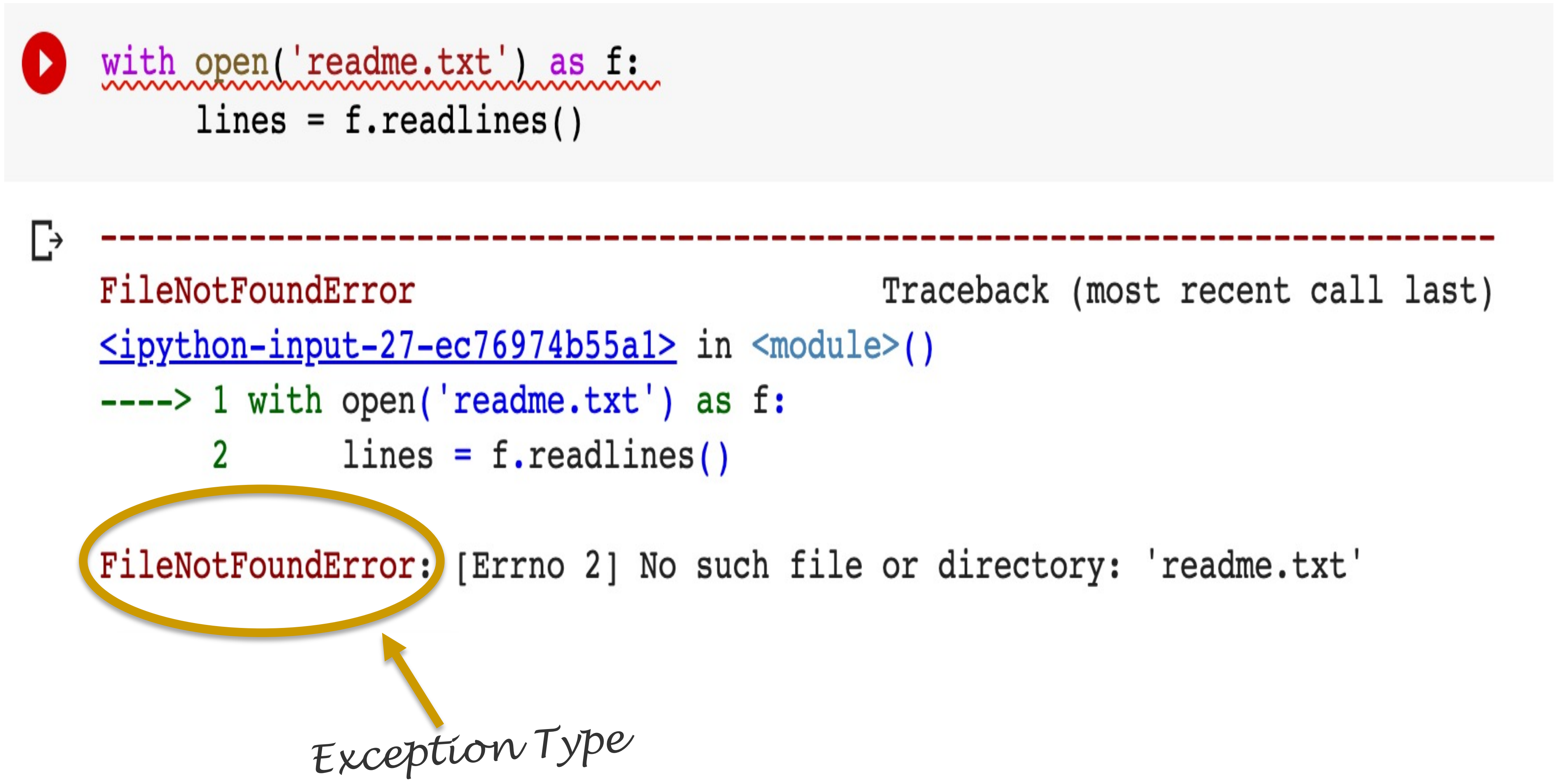
```
1/0
```

```
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
<ipython-input-26-9e1622b385b6> in <module>()
----> 1 1/0

ZeroDivisionError: division by zero
```



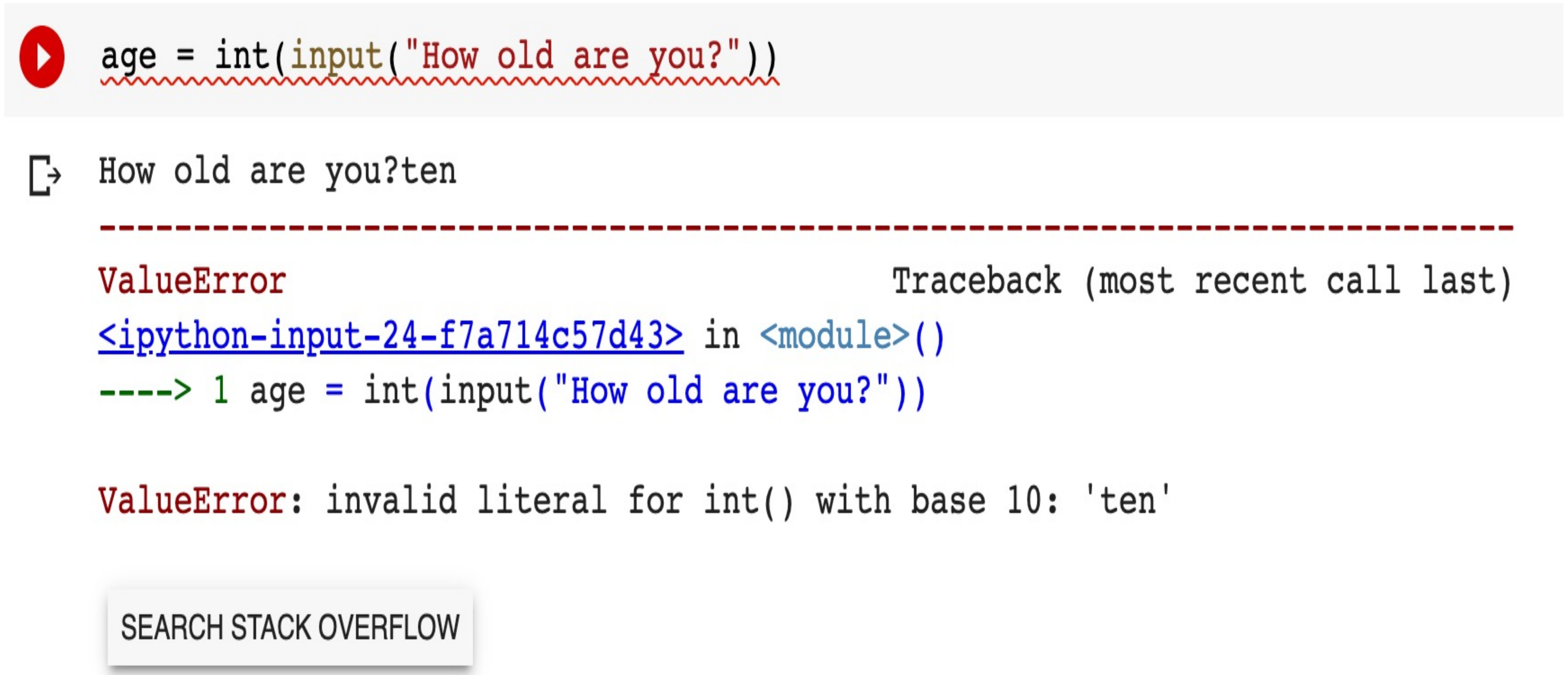
```python
with open('readme.txt') as f:
    lines = f.readlines()
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
<ipython-input-27-ec76974b55a1> in <module>()
----> 1 with open('readme.txt') as f:
      2     lines = f.readlines()

FileNotFoundError: [Errno 2] No such file or directory: 'readme.txt'
```



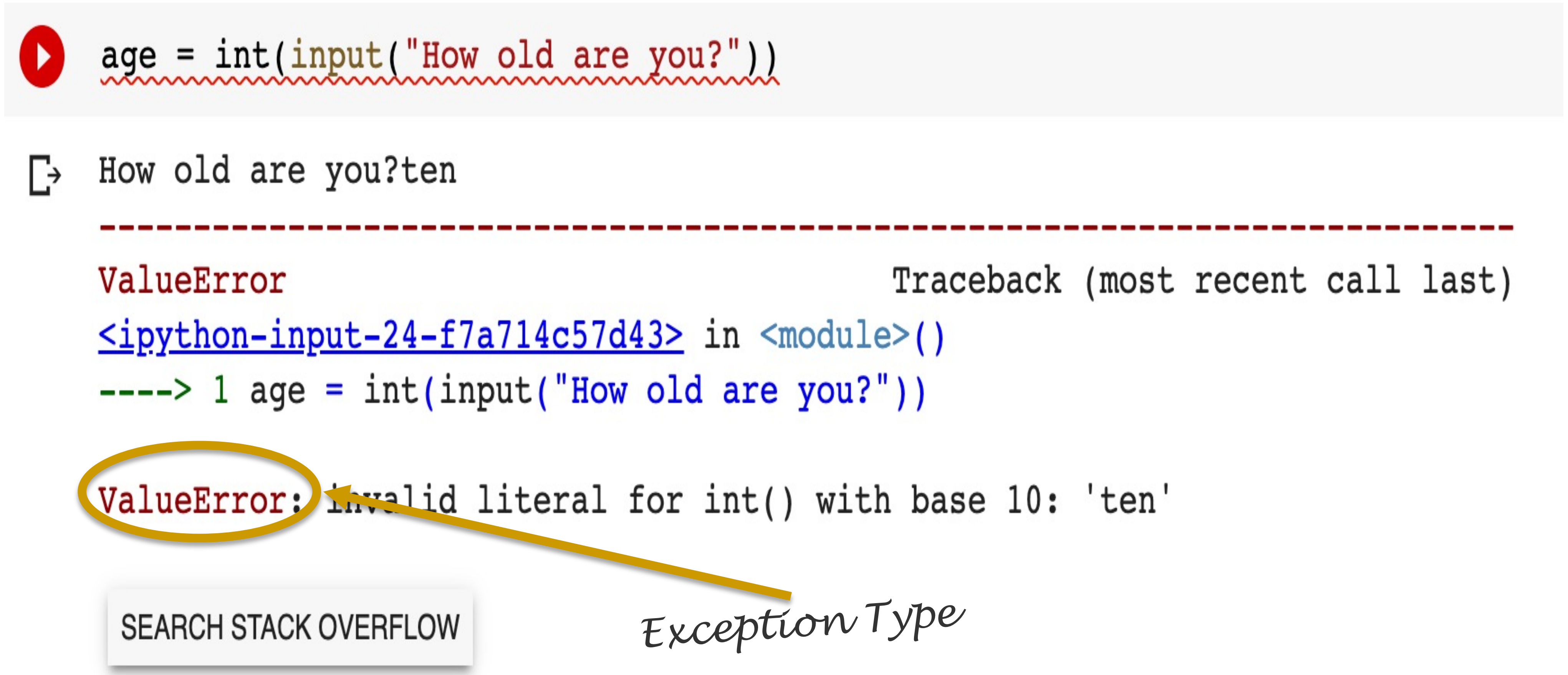
```
with open('readme.txt') as f:
    lines = f.readlines()
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
<ipython-input-27-ec76974b55a1> in <module>()
----> 1 with open('readme.txt') as f:
      2     lines = f.readlines()

FileNotFoundError: [Errno 2] No such file or directory: 'readme.txt'
```



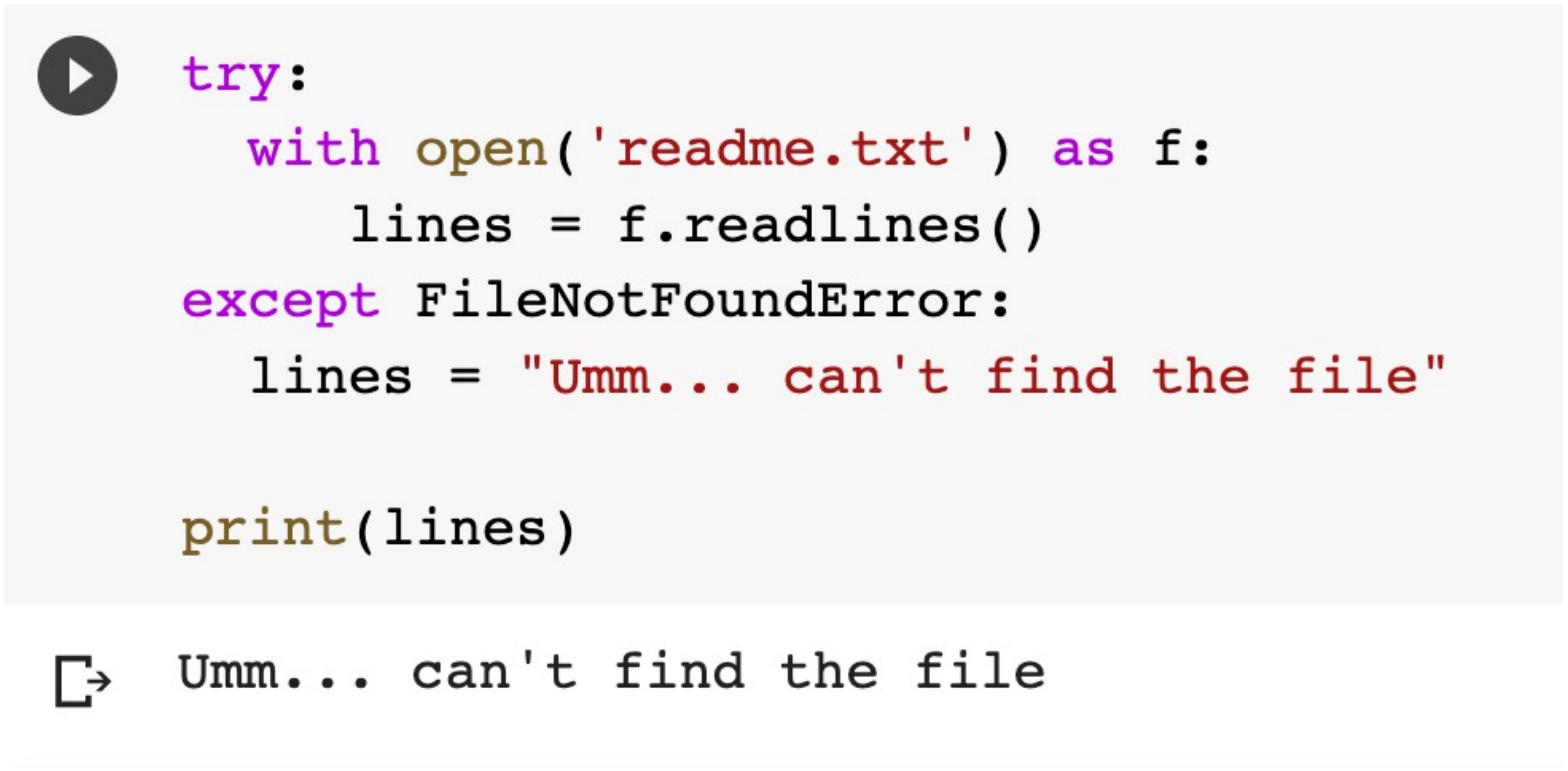
```
age = int(input("How old are you?"))
```

```
How old are you?ten
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-24-f7a714c57d43> in <module>()
----> 1 age = int(input("How old are you?"))

ValueError: invalid literal for int() with base 10: 'ten'
```



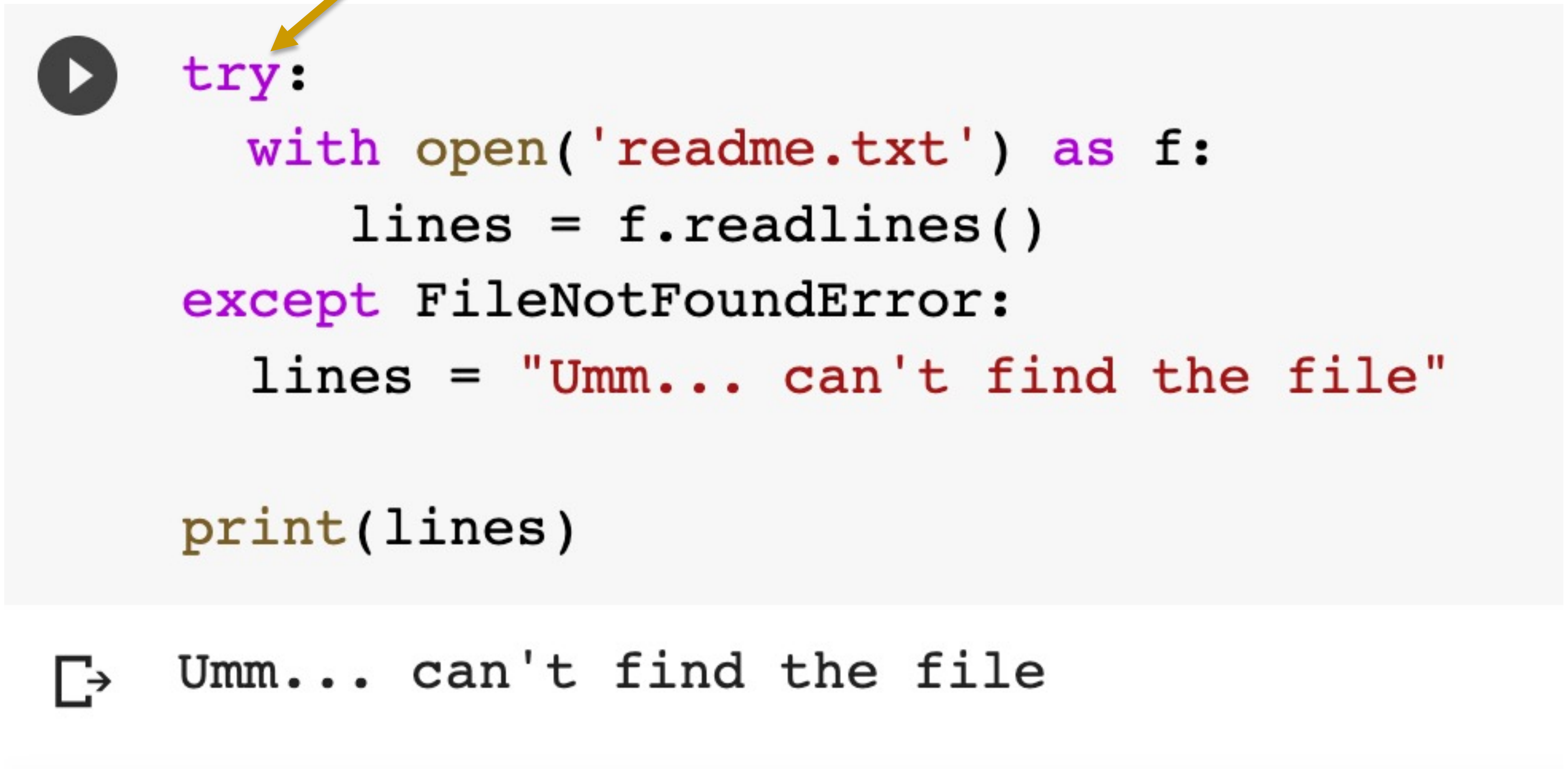
```
age = int(input("How old are you?"))
```

```
How old are you?ten
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-24-f7a714c57d43> in <module>()
----> 1 age = int(input("How old are you?"))

ValueError: invalid literal for int() with base 10: 'ten'
```


```python
try:
  with open('readme.txt') as f:
      lines = f.readlines()
except FileNotFoundError:
  lines = "Umm... can't find the file"

print(lines)
```

```
Umm... can't find the file
```

Create a new block to 'try' the problem code

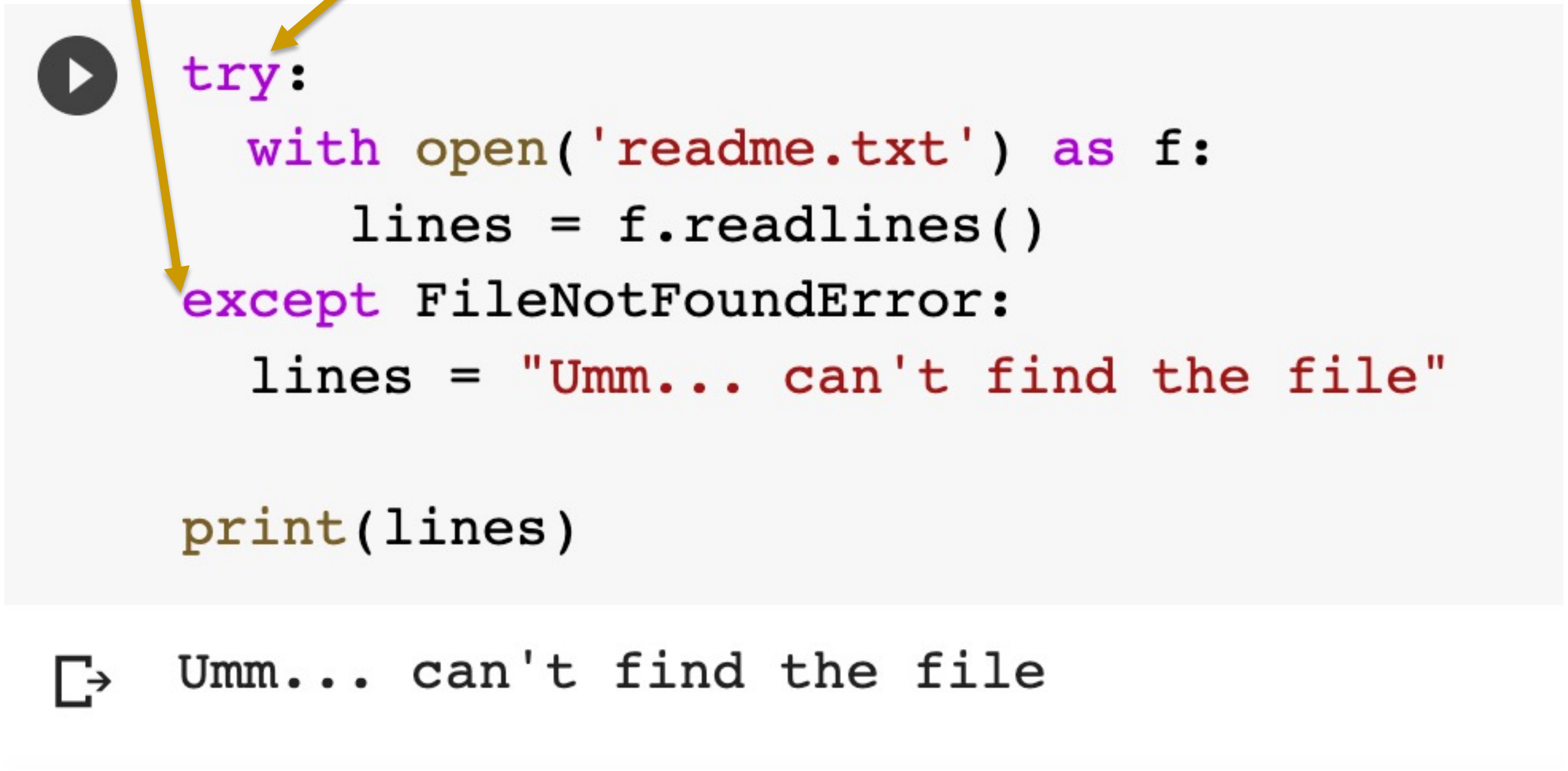


```python
try:
    with open('readme.txt') as f:
        lines = f.readlines()
except FileNotFoundError:
    lines = "Umm... can't find the file"

print(lines)
```

```
Umm... can't find the file
```

Block to run if something goes wrong

Create a new block to 'try' the problem code

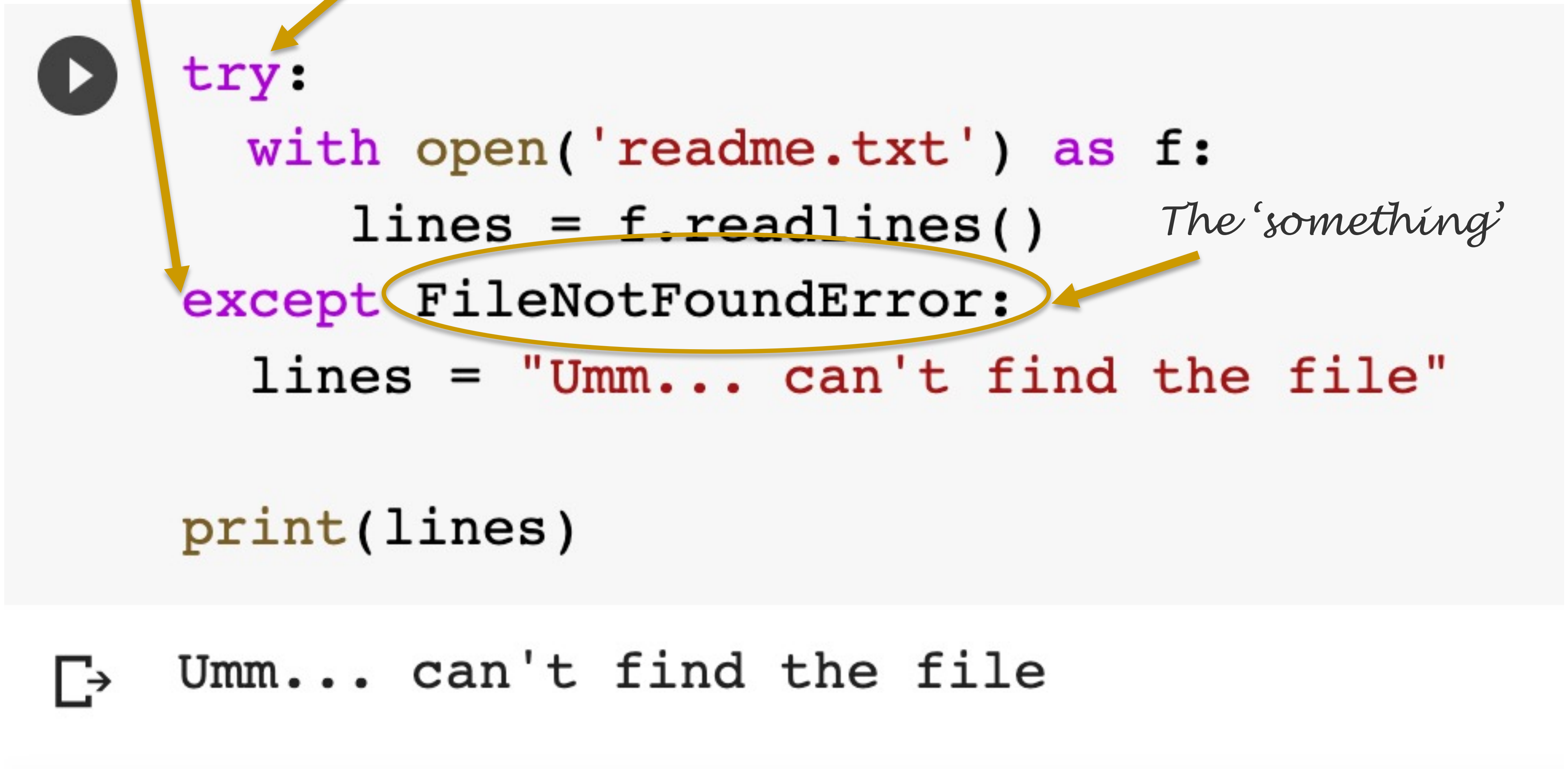


```python
try:
   with open('readme.txt') as f:
      lines = f.readlines()
except FileNotFoundError:
   lines = "Umm... can't find the file"

print(lines)
```

```
Umm... can't find the file
```

Block to run if something goes wrong

Create a new block to ‘try’ the problem code



```
try:
  with open('readme.txt') as f:
    lines = f.readlines()
except FileNotFoundError:
  lines = "Umm... can't find the file"

print(lines)
```

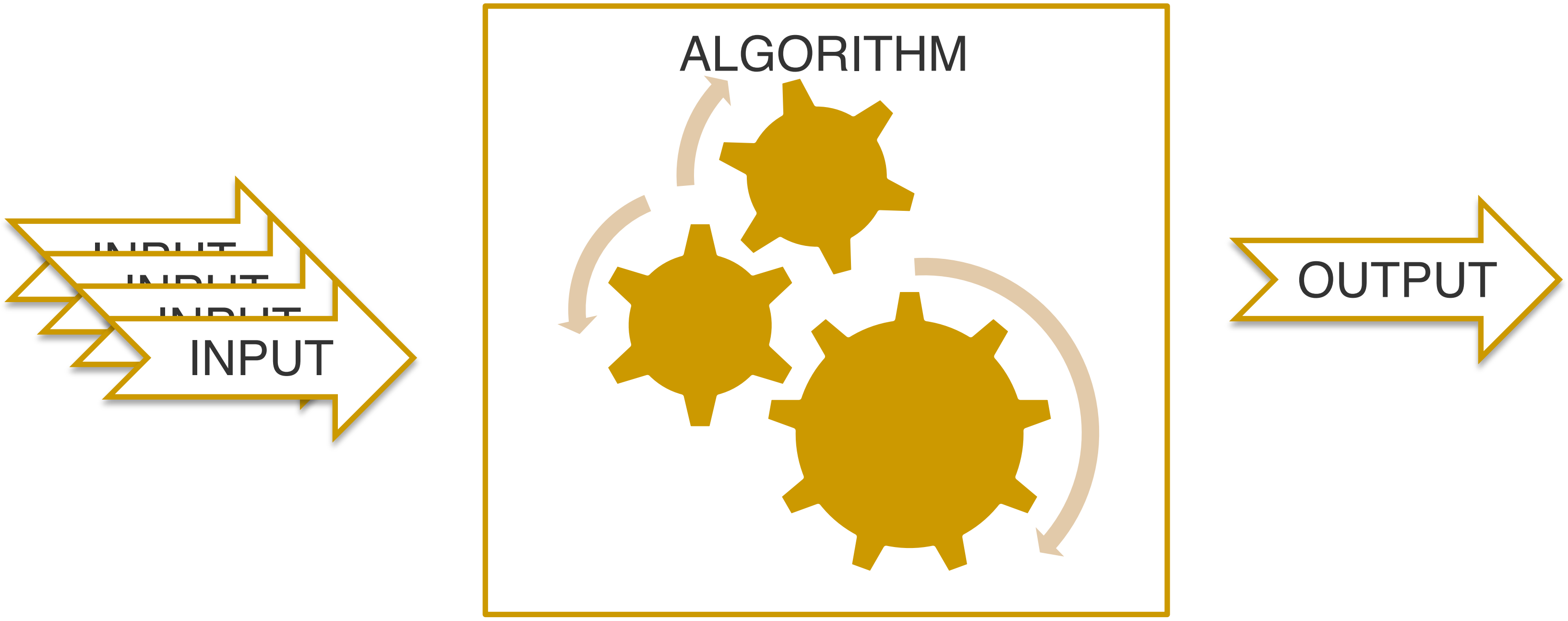
```
Umm... can't find the file
```

The ‘something’

Error Handling

- Exceptions
  - SyntaxError (fix, don’t use try/exception)
  - ZeroDivisionError
  - FileNotFoundError
  - ValueError
  - Many others
- try/except block (simple)
- try/except/else/finally (advanced)

(Update) Input Output Model



Input validation code verifies that user supplied data, such as text from the input() function, is formatted appropriately.

3 Ways

- try/except
- isdigit()
- PyInputsPlus

```
while True:
    try:
        age = int(input('How old are you? '))
        break
    except ValueError:
        print('Please enter a whole number')

print('Your age is: ' + str(age))

#Reference: http://www.easypythondocs.com/validation.html
```

```
while True:
    age = input('How old are you? ')
    if age.isdigit():
        break
    else:
        print('You must enter a valid number')

print('You are ' + str(age))
```

#Reference: http://www.easypythondocs.com/validation.html

PyInputsPlus

- **inputStr()**
- **inputNum()**
- **inputChoice()**
- **inputMenu()**
- **inputDatetime()**
- **inputYesNo()**
- **inputBool()**
- **inputEmail()**
- **inputFilepath()**
- **inputPassword()**

```
import pyinputplus as pyip
response = pyip.inputNum()
```

Input Validation

- Don’t trust input
- try/except
- isdigit()
- Module: pyinputplus

Text Files

- Plain text
- XML
- CSV
- JSON
- Source Code

Binary Files

- Compiled Code
- App data
- Media files

  images

  Audio

  video

```
open(path_to_file, mode)
```

Mode	Description
‘r’	Open a text file for reading text **(default)**
‘w’	Open a text file for writing text
‘a’	Open a text file for appending text

```
with open('readme.txt') as f:
    lines = f.readlines()
```

The default mode is read

```
with open('readme.txt') as f:
    lines = f.readlines()
```

```
with open('readme.txt', 'w') as f:
    f.writelines(lines)
```

Open the file in 'write' mode

```
with open('readme.txt', 'w') as f:
    f.writelines(lines)
```

Working with files

- Binary Files

  use packages (next few weeks)

- Text File

  with keyword

  open()

  read/write/append

Can you

- Get Help with Python
- Handle Exceptions
- Validate Input
- Use Files