I

Loops and Error Handling

Data Science Academy

For Loops

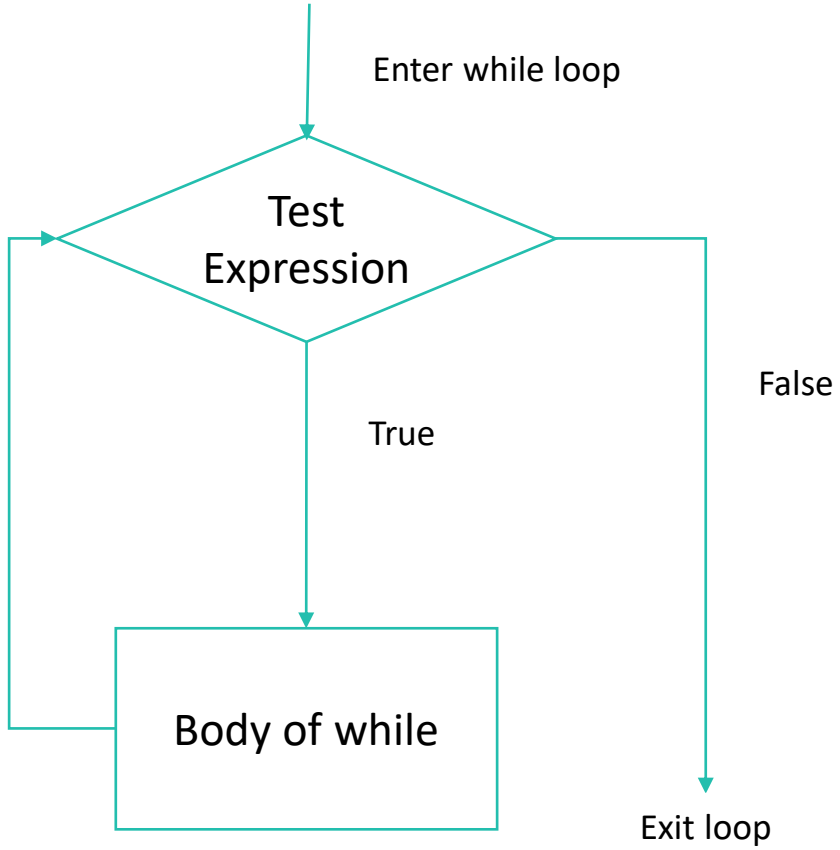
- **Iteration** is a fundamental building block of all programs. It is the ability to execute a certain code repeatedly.
- The list "even" contains all the even numbers from 0 to 20. "for n in even", colon, which would mean *for every* element n in the list "*even*", do the following: print that element.
- he command in the loop body is performed once *for each* element in the *even* list.

```
In [ ]: for n in even :
            body of the loop
```

```
In [ ]: for n in even :
            print (n)
```

While Loops and Incrementing

- The while loop in Python is used to iterate over a block of code as long as the test expression (condition) is true



Errors and Exceptions

- We can make certain mistakes while writing a program that leads to errors when we try to run it. A python program terminates as soon as it encounters an unhandled error. These errors can be broadly classified into two classes:

    - Syntax errors
    - Logical errors (Exceptions)

Syntax Errors

- Error caused by not following the proper structure (syntax) of the language is called **syntax error** or **parsing error**.

```
In  [ ]:  Print ( 0 / 0 ))
```

```
  File "<ipython-input-165-f125ac033bc4>", line 1
    print( 0 / 0 ))
                  ^
SyntaxError: invalid syntax
```

Logical errors (Exceptions)

- Errors that occur at runtime (after passing the syntax test) are called **exceptions** or **logical errors**.
- For instance, they occur when we try to open a file(for reading) that does not exist (FileNotFoundError), try to divide a number by zero (ZeroDivisionError), or try to import a module that does not exist (ImportError).
- Whenever these types of runtime errors occur, Python creates an exception object. If not handled properly, it prints a traceback to that error along with some details about why that error occurred.

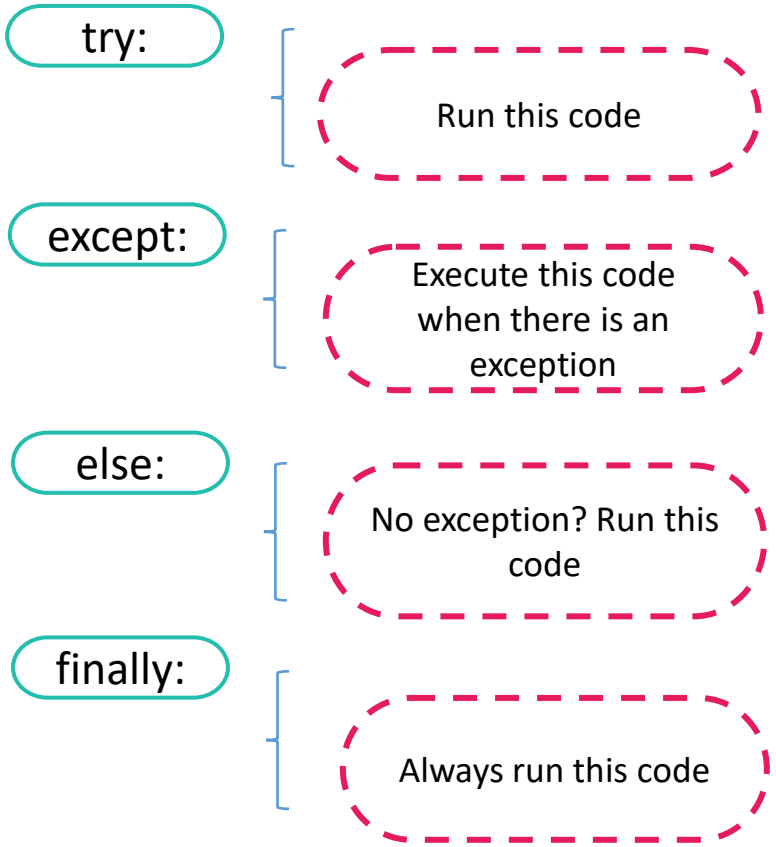
Built-in exceptions

- Illegal operations can raise exceptions. There are plenty of built-in exceptions in Python that are raised when corresponding errors occur.
- We can view all the built-in exceptions using the built-in local() function.

Exception	Cause of Error
AssertionError	Raised when an assert statement fails
AttributeError	Raised when attribute assignment or reference fails
EOFError	Raised when the input() function hits end-of-file condition.
FloatingPointError	Raised when a floating point operation fails
GeneratorExit	Raise when generator's close() method is called
ImportError	Raised when the imported module is not found
IndexError	Raised when the index of a sequence is out of range

Catching exceptions

- In Python, exceptions can be handled using a try statement. The critical operation which can raise an exception is placed inside the try clause. The code that handles the exceptions is written in the except clause.



II

Anonymous Function and Filter

Data Science
Academy

Lambda functions

- In Python, an anonymous function is a function that is defined without a name.
- While normal functions are defined using the def keyword in Python, anonymous functions are defined using the lambda keyword.

```
In [ ]: lambda arguments: expression
```

- Lambda functions can have any number of arguments but only one expression. The expression is evaluated and returned. Lambda functions can be used wherever function objects are required.

- You can apply the function above to an argument by surrounding the function and its argument with parentheses:

```
In [ ]: >>> (lambda x: x + 1)(2)
        3
```

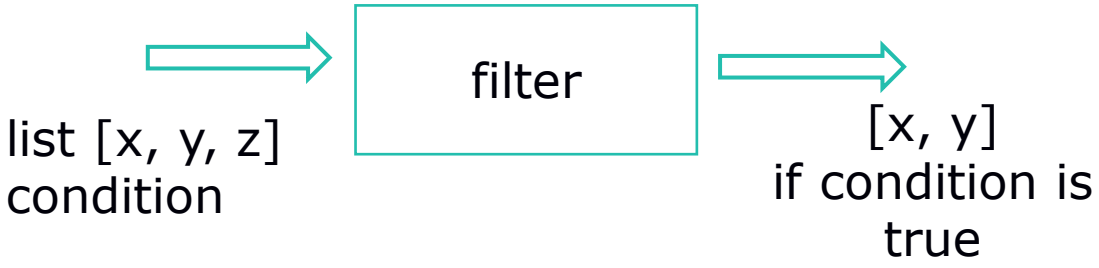
- We use lambda functions when we require a nameless function for a short period of time.
- In Python, we generally use it as an argument to a higher-order function (a function that takes in other functions as arguments). Lambda functions are used along with built-in functions like filter(), map() etc.

Filter

- While map() passes each element in the iterable through a function and returns the result of all elements having passed through the function, filter(), first of all, requires the function to return boolean values (true or false) and then passes each element in the iterable through the function, "filtering" away those that are false. It has the following syntax:

```
In [ ]: filter(function, iterable...)
```

- The following points are to be noted regarding filter():
  - Unlike map(), only one iterable is required.
  - The func argument is required to return a boolean type. If it doesn't, filter simply returns the iterable passed to it. Also, as only one iterable is required, it's implicit that func must only take one argument.
  - Filter passes each element in the iterable through func and returns only the ones that evaluate to true.



III

Zip and Map

Data Science
Academy

Map

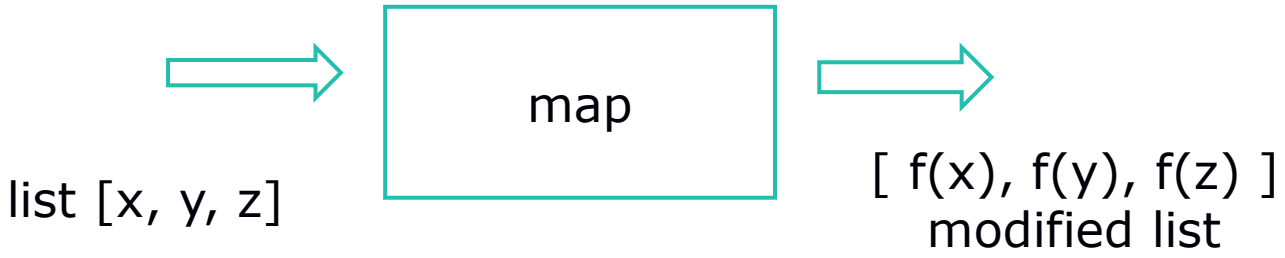
- The map() function in python has the following syntax:

```
In  [ ]:  map (function, iterable, . . .)
```

- Python map function or map data structure implements a given function to each item of an iterable (list, tuple, etc.) and returns a list of the results.

- Parameter Values:
  - function – The function (add, square input, etc) to execute for each item (Required)
  - iterable – A sequence, collection, or an iterator object. You can send as many iterable as you like, just make sure the function has one parameter for each iterable.

- You can pass more than one iterable to the map() function.



Zip

- The zip() function takes iterables (can be zero or more), aggregates them in a tuple, and return it.
  filter(function, iterable...)

- The zip() function returns an iterator of tuples based on the iterable objects.
  - If we do not pass any parameter, zip() returns an empty iterator
  - If a single iterable is passed, zip() returns an iterator of tuples with each tuple having only one element.
  - If multiple iterables are passed, zip() returns an iterator of tuples with each tuple having elements from all the iterables.

  Suppose, two iterables are passed to zip(); one iterable containing three and other containing five elements. Then, the returned iterator will contain three tuples. It's because iterator stops when the shortest iterable is exhausted

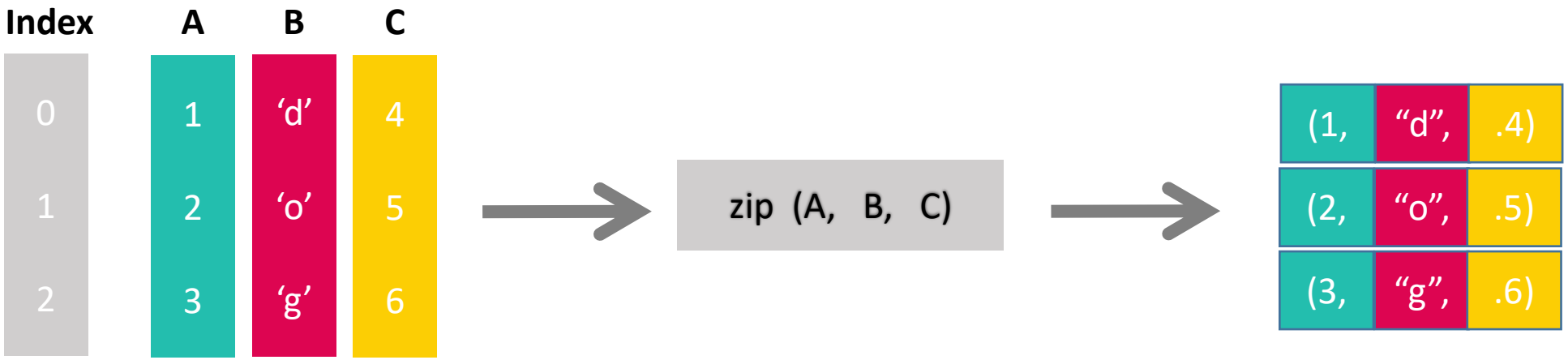
  .

Zip

Understanding the Python zip function

Parallel iteration

Iterables (lists, tuples, dicts, sets, etc.) → Iterator of tuples with i-th element of each



Thank You

Data Science
Academy