The Python Standard Library

Describes the library distributed with Python

Lina Liu Nov. 16 2022

What Python's Standard Library Provide?

1. What python's standard library provide?
    1. Wide range of facilities
    2. built-in modules(written in C) to access to system functionality such as file I/O
    3. Standardized solutions for many problems that occur in everyday programming
    4. Modules designed to abstracting away platforms — neutral APIs
    5. Provides as a collection of packages
    6. Collections of packages: python package index website
        1. https://pypi.org/

The Python Standard Library

Introduction

1. What Python language core defines?
    1. “Core” of language is data types like lists, dictionary, set, tuple modules
    2. Defines the form of literals and places some constraint on their semantics
    3. Built-in functions and exceptions without need of an import statement
2. Bulk of the library introduction
    1. Modules written in C and built in to the Python interpreter.
    2. Modules written in Python and imported in source form
    3. Modules provide interfaces that are highly specific to Python
        1. Printing a stack trace
    4. Modules provide interfaces that are specific to particular operating system
        1. Access to specific hardware
    5. Modules provide interfaces that are specific to a particular application domain
        1. Like for World Wide Web
    6. Some modules are available in all versions and ports of python
    7. Some modules are only available when the underlying system support or require them
    8. Some modules are only available when a particular configuration option was chosen at the time when python was compiled and installed.

The Python Standard Library

Contents

The Python Standard Library

Introduction

1. How the manual organized?
    1. From the inside out organized
    2. First describes the built-in functions
    3. Second describe “core” data types and exceptions
    4. Third describe the modules
2. How to read the Python Standard Library?
    1. Just browse the table of contents
    2. Look for a specific function, module in the index
    3. Better to start with Built-in Functions
    4. It’s kind of dictionary lookup tool, instead of a novel.

Built-in functions

1. bin(number)
2. format(number, ‘#b’) format(number, ‘b’)
3. dir()
    1. If the object is module object, the list contains the names of the module’s attributes
    2. If the object is a type or class object, the list contains the names of its attributes, and recursively of the attributes of its bases.
    3. Otherwise, the list contains the objects’s attributes’ names, the names of its class’s attributes, and recursively of the attributes of its class’s base classes.
4. enumerate(iterable, start=0)
    1. Return an enumerate object
    2. Iterable must be a sequence, an iterator, or some other object which supports iteration
    3. The __next__() method of the iterator returned by enumerate() returns a tuple containing a count and the values obtained from iterating over iterable.
5. eval(expression,globals=None, locals=None)
6. float(string)
7. hex()
8. input()
9. iter(object, sentinel, /): return an iterator object, you can define __iter__() and __next__(self) in you custom object
10. open(file, mode=‘r’, buffering=-1, encoding=None, errors=None, newline=None, closefd=True, opener=None)
11. zip(*iterables, strict=False): iterate over several iterables in parallel, producing tuples with an item form each one.

```python
# So funny, NotImplementedError: dir_fd unavailable on this platform!!!!!!!
import os
dir_fd = os.open('/Users/lina/code/core-python/', os.O_RDONLY)
def operner(path, flags):
    print('not implemented!')
    #return os.open(path, flags, dir_fd=dir_fd)

#with open('test_file.txt', 'a', opener=operner) as f:
with open('test_file.txt', 'a') as f:
    print('This will be written to xxxx/test_file.txt', file=f)

os.close(dir_fd)
```

Character	Meaning
`'r'`	open for reading (default)
`'w'`	open for writing, truncating the file first
`'x'`	open for exclusive creation, failing if the file already exists
`'a'`	open for writing, appending to the end of file if it exists
`'b'`	binary mode
`'t'`	text mode (default)
`'+'`	open for updating (reading and writing)