Outline: Python Standard Library

Seminar on Selected Tools Week 1

Yifei Wang pppppass

February 26, 2018

Index

datetime I

1. Available types
   - date
   - time
   - datetime
   - tzinfo
   - timezone
2. **timedelta** objects
   - class `datetime`.**timedalta**(*days=0, seconds=0, microseconds=0, milliseconds=0, minutes=0, hours=0, weeks=0*)
   - Unique representation: *days*, *seconds* and *microseconds*
   - Special supported operation
     - abs(t)
     - str(t)

datetime II

3. *date* Objects
   - class `datetime`.**date**(*year, month, day*)
   - All arguments are required.
   - Class methods
     - replace(year=self.year, month=self.month, day=self.day)
     - toordinal()
     - isoweekdat()
     - isocalendar()
     - isoformat()

datetime III

4. *time* Objects
   - class `datetime`.**time**(textithour=0, minute=0, second=0, microsecond=0, tzinfo=None, *, fold=0)
   - All arguments are optional.
   - Instance methods:
     - replace(hour=self.hour, minute=self.minute, second=self.second, microsecond=self.microsecond, tzinfo=self.tzinfo, * fold=0)
     - isoformat(timespec='auto')

math I

1. Number-theoretic and representation functions
   - floor(x)
   - fabs(x)
   - factorial(x)
   - fmod(x, y)
   - fsum($iterable$)
   - gcd(a, b)
   - isclose($a, b, *, rel\_tol = 1e - 09, abs\_tol = 0.0$)
2. Power and logarithmic functions
   - exp(x)
   - log($x[, a]$)
   - pow(x, y)

math II

3. Trigonometric and Hyperbolic functions
   - acos(x)
   - cos(x)
   - hypot(x, y)
   - acosh(x)
   - cosh(x)
4. Constants
   - pi
   - e
   - inf
   - nan

random

- random()
- randrange($m[, n, [d]]$)
- randint(m, n)
- choice(s)
- seed($[n]$)
- shuffle(x)
- sample($popluation, k$)

pickle

1. Functions
   - dump(obj, file, protocol=None, *, fix_imports=True)
   - dumps(obj, protocol=None, *, fix_imports=True)
   - load(file, *, fix_imports=True, encoding="ASCII", errors="strict")
   - loads(bytes_object, *, fix_imports=True, encoding="ASCII", errors="strict")
2. Classes
   - class pickle.Pickler(file, protocol=None, *, fix_imports=True)
     dump(obj)
   - class pickle.Unpickler(file, *, fix_imports=True, encoding="ASCII", errors="strict")
     load()

json

1. dump(obj,fp)(obj, fp, *, skipkeys=False, ensure_ascii=True, check_circular=True, allow_nan=True, cls=None, indent=None, separators=None, default=None, sort_keys=False, **kw)
2. dumps(obj, *, skipkeys=False, ensure_ascii=True, check_circular=True, allow_nan=True, cls=None, indent=None, separators=None, default=None, sort_keys=False, **kw)
3. load(fp, *, cls=None, object_hook=None, parse_float=None, parse_int=None, parse_constant=None, object_pairs_hook=None, **kw)
4. loads(s, *, encoding=None, cls=None, object_hook=None, parse_float=None, parse_int=None, parse_constant=None, object_pairs_hook=None, **kw)

collections I

1. ChainMap objects
   - maps
   - new_child($m = None$)
   - parents
2. Counter objects
   - elements()
   - most
   - subtract([*iterable*])

collections II

3. deque objects
   - append/appendleft(x)
   - clear()
   - copy()
   - count(x)
   - extend/extendleft(*iterable*)
   - index(x[, *start*[, *stop*]])
   - insert(i, x)
   - pop()/popleft()
   - remove(*value*)
   - reverse()
   - rotate(n)

collections III

4 defaultdict objects

5 nametuple()

6 OrderedDict()

re I

1. Special characters:
   - .
   - +
   - ?
   - {m}
   - {m,n}
   - \
     - \d
     - \s
     - \S
     - \w
     - \W

re II

2. Module contents:
   - compile($pattern$, $flags = 0$)
   - search($pattern$, $string$, $flags = 0$)
   - match($pattern$, $string$, $flags = 0$)
   - split($pattern$, $string$, $maxsplit = 0$, $flags = 0$)

itertools

1. Infinite iterators:
   - count(*start*, [*step*])
   - cycle(*p*)
   - repeat(*elem*[, *n*])
2. Itertool functions:
   - accumulate(*iterable*[, *func*])
   - chain($*$*iterables*)
   - groupby(*iterable*, *key* = *None*)

abc

- Declaration:
  `def AbstractBaseClass(object, metaclass=ABCMeta)`
- Register as an abstract method: `@abstractmethod`

abc

- Class `Decimal` and class `Fraction`
- Many operators and functions are implemented
- Can be used for high precision (maybe symbolic) computation (And maybe for OI problems)

argparse

- Basic usage: `ArgumentParser`, `add_argument`, `parse_args`
- Key options: `key`, `nargs`
- Default options: `default`, `const`

logging

- Get `logger` instance: `getLogger`, root logger
- Log formatter: `Formatter`, `setFormatter`
- Log handlers: `FileHandler`, `StreamHandler`
- Manipulate handlers: `addHandler`, `removeHandler`
- Several levels: `debug`, `info`, `warning`, `error`, `critical`