EECS 1210 Introduction to Programming in Python
Spring Semester 2024
Prof. Pai H. Chou

Self-Check 13

Answer the following questions to check your understanding of your material. Expect the same kind of questions to show up on your tests. This self check is for Python Standard Library part 1.

1. Definitions and Short Answers - functions

1. If a function is **built-in**, do you have to **import** it first, or can you just call it without importing?
   Just call it without importing
2. If a function is in the **standard library**, do you have to **import** it first, or can you just call it without importing?
   只要不是built-in 就應該需要Import 它
3. Which of the following are valid calls to the eval() function, and what are their return values?
   a. eval(2 + 3)
      TypeError: eval() arg 1 must be a string, bytes or code object
   b. eval('2 + 3')
      5
   c. eval(len("hello"))
      TypeError: eval() arg 1 must be a string, bytes or code object
   d. eval('len("hello")')
      5
   e. eval('hello world')
      SyntaxError: unexpected EOF while parsing
   f. eval('"hello world"')
      'hello world'
4. Which of the following are valid calls to the eval() function, and what are their return values?
   a. a = 5
      eval('a + 3')
      8
   b. a = 5
      eval('a + 3', {'a': 7})
      10
   c. a = 5
      eval('a + 3', {})
      NameError: name 'a' is not defined
5. Is the following a valid call to exec() function and what is the results?

a. exec('def hello(s):\n print(f"hello {s}")\n')
hello('Mike')
[valid]
hello Mike #result

6. When your Python program first starts,

```
>>> x = 3
>>> dir()
['__annotations__', '__builtins__', '__doc__', '__loader__', '__name__', '__package__', '__spec__', 'x
']
```

which shows that the name 'x' has been added to the global name space. However,

```
>>> max(x, 5)
5
>>>
```

However, the name max is not in the global name space as shown in dir(). Which of the following is a correct explanation of how Python knows max is the name of a function that you can call?

a. max is a **keyword** in Python
b. max is defined in the **global** name space but is just **hidden**
c. max is defined in the __builtins__ name space, which is searched after the global name space
d. max is defined in a package in the **standard library** and must be **imported** before it can be called.

7. If you do the following

```
>>> max, min = min, max
>>> L = [1, 7, 3, 4, 6]
>>> max(L), min(L)
```

What do you get? (1,7)

8. If you do the following

```
>>> L = [1, 7, 3, 4, 6]
>>> len = 'a'
>>> len(L)
```

a. What do you get by len(L)? or do you get an error?
you get an error, because len is now defined to be the string 'a' and you can't call it.
b. if the built-in len() function has been redefined to 'a', can you still get the original definition back? If so, how? can the original len() be called?
len = __builtins__.len.

9. The datetime module contains the following classes
datetime

date
time
timedelta

a. Do you need to do an import statement before you can use any of the four classes above?
yes

b. If you see the statement
import datetime
Does datetime refer to the **module** or **class**?
>>> import datetime
>>> type(datetime)
<class 'module'>

c. if you see the statement
from datetime import *
What is the meaning of * ?
After the import statement, does datetime refer to the **module** or **class**?
>>> from datetime import *
>>> type(datetime)
<class 'type'>

d. If you want to import the "datetime **class within** the datetime **module**" without importing t he other classes (date, time, deltatime), what import statement should you use?
from datetime import datetime

e. The constructor for the datetime class takes three required parameters: *year*, *day*, and *hour*. What is the Python code for instantiating an object of datetime class with year=2019, mon th=12, day=10, and assign it to the identifier x? Include also the statement for import.
import datetime
x = datetime.datetime(2019, 12, 10)

could also do
from datetime import *
x = datetime(2019, 12, 10)

f. The documentation says datetime.now is a **class method** that takes no parameter. How do you call it on the datetime class? If x is an instance of datetime from the previous questio n, are you allowed to call x.now()?
from datetime import datetime
n = datetime.now()
yes , It is allowed to call x.now()

10. **Operator overloading** is supported on datetime and deltatime classes. Indicate the combinations su pported by filling in the result type; or indicate if the combination is not supported.

class	operator	class	result class
timedelta	+	timedelta	timedelta
timedelta	-	timedelta	timedelta
timedelta	*	int	timedelta
datetime	+	datetime	TypeError: unsupported operand type(s) for +: 'datetime.datetime' and 'datetime.datetime'
datetime	-	datetime	timedelta
datetime	+	timedelta	datetime
datetime	-	timedelta	datetime

11. In the calendar module,
    a. what is the difference between the TextCalendar and the HTMLCalendar class?
       Text出來的是純文字
       HTML出來的是網頁語法,可以嵌入在網站上
    b. in the TextCalendar class, what is the difference between the prmonth() and formatmonth() methods?
       prmonth印出來的會是像月曆的格式一樣
       formatmonth就是字串的形式
12. In the namedtuple class in the collections module,
    >>> from collections import namedtuple
    >>> Point = namedtuple('Point', ['x', 'y'])
    a. What is the type of Point? Is it an instance of namedtuple class, or is a class? <class 'type'> it is a class
    b. To continue with the example above, the next line is
       >>> p = Point(2, 3)
       What kind of object is p? Is it an instance of namedtuple class? an instance of Point class? an instance of tuple? Is it mutable?
       <class '__main__.Point'>
       it is an instance of Point
       it is an instance of tuple
       (Point is subclass of tuple)
       it is immutable.
13. In the Counter class in the collections module,
    >>> from collections import Counter
    >>> c = Counter(['dog','cat','dog','cow','dog','cat'])

```
>>> c
```

a. What do you expect to see?
   Counter({'dog': 3, 'cat': 2, 'cow': 1})
b. How do you find the value with the most occurrences?
   dog
c. How do you find the value with the least occurrences?
   cow

14. Consider the overloaded operators defined by Counter class in the collections module, what is the value you expect from the expressions?

a	op	b	value
Counter('abacus')	+	Counter('aba')	Counter({'a': 4, 'b': 2, 'c': 1, 'u': 1, 's': 1})
Counter('abacus')	-	Counter('ada')	Counter({'c': 1, 'u': 1, 's': 1})
Counter('ada')	-	Counter('abacus')	Counter({'d':1})
Counter('dust')	&	Counter('rust')	Counter({'u': 1, 's': 1, 't': 1})
Counter('dust')	|	Counter('rust')	Counter({'d': 1, 'u': 1, 's': 1, 't': 1, 'r': 1})

15. In the collections.abc module for **abstract base classes**, some such base classes include Container, Hashable, Iterable, Iterator, …, Sequence, MutableSequence, etc.
    a. Can you **instantiate** an object from one of these abc's? for example,
    ```
    from collections import abc
    x = abc.Iterable()
    ?
    ```
    TypeError: Can't instantiate abstract class Iterable with abstract methods __iter__
    b. Can you test if an object is **an instance of** one of these abc's? for example,
    ```
    x = 'hello'
    if isinstance(x, abc.Iterable):
        print('str is iterable')
    else:
        print('str is not iterable')
    ```
    str is iterable

16. In the types module, a number of classes are also defined, including FunctionType, LambdaType, GeneratorType, MethodType, BuiltinFunctionType, ModuleType, etc.

a. Are these classes in types module also abstract base classes like those defined in collection s.abc module?
這些是標準函式庫的內容，會有實例，跟abc不太像?

b. Are you expected to use these classes for instantiation by calling their constructors? If no t, why not? For example, is the following the expected usage

```
>>> import types
>>> f = types.FunctionType()
```

這樣的話會產生錯誤訊息

TypeError: function() missing required argument 'code' (pos 1)

也就是說要傳code內容給這個function物件

還是希望用def標準寫法會比較好

17. What is the purpose of an **enumerated type** as in the Enum class in the enum module? Consider th e code

```
1 from enum import Enum
2 Animal = Enum('Animal', ['ANT', 'BEE', 'CAT', 'DOG'])
```

a. What is the value of Animal(1)? Animal(3)?

```
>>> Animal(1)
<Animal.ANT: 1>
>>> Animal(3)
<Animal.CAT: 3>
```

b. What is the value of Animal.BEE?

```
>>> Animal.BEE
<Animal.BEE: 2>
```

c. What is the value of str(Animal['CAT'])

```
>>> str(Animal['CAT'])
'Animal.CAT'
```

d. What is the value of Animal.DOG > Animal.ANT?

TypeError: '>' not supported between instances of 'Animal' and 'Animal'

不支援大於小於只支援等於不等於

18. What is the difference between the built-in float type and Decimal class in the decimal module?

a. What is the value of

1.1 + 2.2 == 3.3 ?

False

b. Assuming you have from decimal import Decimal, what is the value of

Decimal('1.1') + Decimal('2.2') == Decimal('3.3')

?

True

19. Assume you have from fractions import Fraction, what is the value of

Fraction(16, -10)

?

Fraction(-8, 5)

20. In the random module,
    a. What are the possible values of
       random.randrange(10)
       ? 0~9
    b. What are possible values of
       random.choice(['win', 'lose', 'draw'])
       ? ‘win’ , ‘lose’ , ‘draw’
    c. What is the purpose of a random seed? If you have
       import random
       r = random.Random()
       r.seed(100)
       x = r.randint(1, 100)
       r.seed(100)
       y = r.randint(1, 100)
       is x == y?
       yes
21. In the itertools module,
    a. there is a class named count, and you can use it like
       >>> c = itertools.count(10)
       >>> next(c)
       10
       >>> next(c)
       11
       >>> next(c)
       12
       Why is this class useful, and why can't it be done with the built-in range()?
       range一定要有開始與結束 但count可以無限成長下去
    b. There is another class named cycle, and you can use it like
       >>> cy = itertools.cycle(['a', 'b', 'c'])
       >>> next(cy)
       'a'
       >>> next(cy)
       'b'
       >>> next(cy)
       'c'
       >>> next(cy)
       'a'
       >>> next(cy)
       'b'
       Why is this class useful, and why can't it be done with the built-in range()?
       cycle可以無限一直循環

c. There is another class named zip_longest. Example use is

```
>>> list(itertools.zip_longest('ABCD', 'WXY', '12', fillvalue='-'))
[('A', 'W', '1'), ('B', 'X', '2'), ('C', 'Y', '-'), ('D', '-', '-')]
```

Explain how this can be useful for adding two polynomial functions.
他如果沒有長度相同可以指定空的位置預設fillvalue,十分方便

2. Programming

1. (Difficulty: ★★☆☆☆) Write a "rock, paper, scissors" game using the random module.

```
$ python3 rps.py
rock, paper, scissors? rock
I am also rock - tied!
rock, paper, scissors? paper
I am rock - I lose!
rock, paper, scissors? scissors
I am rock - you lose!
rock, paper, scissors? rabbit
rabit is invalid - try again? quit
bye
$
```

```
import random

def battle(your_sign, my_sign):
   rock_dict = {'rock' : 'tied!', 'paper' : 'I lose!', 'scissors' : 'you lose!'}
   paper_dict = {'rock' : 'you lose!', 'paper' : 'tied!', 'scissors' : 'I lose!'}
   scissors_dict = {'rock' : 'I lose!', 'paper' : 'you lose!', 'scissors' : 'tied!'}

    battle_dict = {'rock' : rock_dict, 'paper' : paper_dict, 'scissors' : scissors_dict}

    s = ''
    s += 'I am '
    if your_sign == my_sign:
        s += 'also '
    s += my_sign
    s += ' - '
    s += battle_dict[my_sign][your_sign]

    print(s)
```

```
flag = False
while True:
    if(not flag):
        sign = input('rock, paper, scissors? ').split()[0]
        flag = False
    if sign == 'quit':
        print('bye')
        break
    if sign in {'rock', 'paper', 'scissors'}:
        AI_sign = random.choice(['rock', 'paper', 'scissors'])
        battle(sign, AI_sign)
        flag = False
    else:
        sign = input(f'{sign} is invalid - try again? ').split()[0]
        flag = True
```

2. (Difficulty: ★★★☆☆) Define a Matrix class to represent numbers as a two-dimensional array.

The constructor for the matrix is a list of lists of numbers. A 3x3 matrix

1	2	3
4	5	6
7	8	9

would be constructed as
M = Matrix([[1, 2, 3], [4, 5, 6], [7, 8, 9]])

Define the following methods

```
class Matrix:
    def __init__(self, data):
        # data is the list of lists of value
            self._rows_list = data
    def row(self, r):
        # return the r-th row in the form of a list
        # r is from 0.. number of rows
        # in the exmaple above, M.row(1) would return
        # [4, 5, 6]
            return self._rows_list[r]
    def column(self, c):
```

```
    # retun a the c-th column in the form of a list.
    # in the example above, M.column(2) would return
    # [3, 6, 9]
    result = []
    for index in self._rows_list:
        result.append(index[c])
    return result


@property
def nrows(self):
    # return the number of rows
        return len(self._rows_list)
@property
def ncolumns(self):
    # return the number of columns
        return len(self._rows_list[0])
def __getitem__(self, ij):
    # return the matrix element at ij, where ij is a tuple
    # for the (row, column).  For example above,
    #
        return self._rows_list[ij[0]][ij[1]]
def __setitem__(self, ij, val):
    # assign the value to the matrix as ij, where ij is
    # a tuple for (row, column)
        self._rows_list[ij[0]][ij[1]] = val

def transpose(self):
    # return a new Matrix whose content is same as this
    # Matrix except the row and column positions are
    # switched.  In the example above,
    # M.transpose() would return
    # Matrix([[1, 4, 7], [2, 5, 8], [3, 6, 9]])
    # Note: use zip() to do the transpose
        zipped = list(zip(*self._rows_list))
    for c in zipped:
        c = list(c)
    return Matrix(zipped)

def randomize(self):
    # return another matrix whose content is the same as
```

```
        # this matrix except their positions are randomized.
            import random
        all_list = []
        for row in self._rows_list:
            for element in row:
                all_list.append(element)
        random.shuffle(all_list)
        count = 0
        for row_i,row_v in enumerate(self._rows_list):
            for i,v in enumerate(row_v):
                self._rows_list[row_i][i] = all_list[count]
                count += 1

    def __matmul__(self, other):
            # return the matrix product for the two matrices A B
        # p[i,j] = sum(A[i,k] * B[k,j]) for 0 <= k <=
                    A.ncolumns where A.ncolumns must be == B.nrows
            if self.nrows != other.ncolumns:
            raise ValueError("you can't do that.")
        product_list = []

        for i in range(self.nrows):
            product_list.append([])
            for j in range(other.ncolumns):
                product_list[i].append(0)
                for k in range(self.ncolumns):
                    product_list[i][j] += self._rows_list[i][k] * other._rows_list[k][j]
        return Matrix(product_list)
```

3. (Difficulty: ★★★★☆) Write a function that can take a variable number of parameters to make a postfix calculator for dates. This is similar to the postcalc example from HW8, except you work on dates instead of numbers.

   The parameter list consists of either the operands or the operators. An operand is either a date strin g or a date-delta string and is pushed on the stack. An operator is a string that indicates the action to take. A binary arithmetic operator pops the top two elements from the stack and pushes back th e result.

argument	action
date(year, month, day)	push date onto stack

days(d)	push days delta onto the stack
weeks(w)	push weeks delta onto the stack
months(m)	push months delta onto the stack
'today' 'tomorrow' 'yesterday'	push today's date, tomorrow's date, or yesterda y's date onto the stack
'add'	A = pop(); B = pop(); push(A+B)
'sub'	A= pop(); B = pop(), push(A-B)
'swap'	A=pop(); B=pop(); push(A); push(B)

Write a stack-style date-time calculator function called datecalc. Here is an example

```
$ python3 datecalc.py
>>> datecalc('today', 'tomorrow', 'yesterday', daydelta(4))
[date(2019, 12, 3), date(2019, 12, 4), date(2019, 12, 2), days(4)]
>>> datecalc('today', 'tomorrow', 'yesterday', days(4), 'add')
[date(2019, 12, 3), date(2019, 12, 4), date(2019, 12, 06)]
>>> datecalc('today', months(2), 'add')
[date(2020, 02, 03)]
>>> datecalc('today', date(2019, 12, 10), date(2019, 12, 20), 'sub', 'add')
[date(2019, 12, 13)]
>>> datecalc('today', weeks(2), 'add', months(2), 'swap', 'sub')
[date(2019, 10, 17)]
```

Hint:

Obviously, you should take advantage of the datetime module to do as much of the work as possibl e. You will need to define your own classes

days
weeks
months

The datetime.timedelta class can be the base class for your own days class and weeks class. The d ays class would simply define a constructor that passes the days parameter to the base class; the we eks class is similar except it is simply in units of 7 days. The overloaded operators can be inherite d directly from the timedelta class. You also need to define your own __repr__ special method for these two classes so their values can be displayed accordingly.

Your months class would be your own class. The reason is that the number of days depends on the actual date. So, you can't simply say 2 months is 60 days. Instead, when you do date + months or date - months, you operate on the month (and maybe year) field and return a newly constructed dat e object.

Your months class needs to define the special methods

```
__add__(self, RHS)  # if RHS is date, return new date;
                    # if RHS is months, return sum months
__sub__(self, RHS)  # RHS can only be months;
                    # => return date constructor call with
                    #    updated month value
                    # all other types => type error
__radd__(self, LHS) # simply return self + LHS
                    # and LHS can only be date.
__rsub__(self, LHS) # LHS can only be months;
                    # => return date constructor call with
                    #    updated month value
```

Once these classes are all working, then put your code into the loop structure as in the postcalc. Y ou also need to update the string comparison so that the strings such as yesterday, today, and tomor row get mapped to the respective date object.