2python

July 15, 2023

[ ]:
```
#
L = ['Michael', 'Sarah', 'Tracy', 'Bob', 'Jack']
print(L[-2:])
L = list(range(100))
print(L[::5])
T = (0,0,1,5,1,4)[:3]
T
s = 'hello'[1:3]
s
s=r'let`ting go'[::-1]
s
```

[ ]:
```
#
practice = "                          "
target = "   "
start = practice.find(target)
end = start+len(target)
practice[start:end]
```

[ ]:
```
#          collections.abc    Iterable
from collections.abc import Iterable
isinstance('abc',Iterable)
isinstance((1,3,6),Iterable)
isinstance(123,Iterable)
#Python   enumerate      list    -
for i,val in enumerate(["c","d","e"]):
    print(i,val)
for x,y in [(1,3),(3,4),(5,6)]:
    print(x,y)
for x,y in {"name":'jack','age':20,'sex':1}.items():
    print(x,y)
```

[ ]:
```
#        list       tuple
def min_max(ls):
    max = ls[0]
    min = ls[0]
    for i in ls:
        if i>max:
```

```
            max = i
        if i<min:
            min = i
    return (min,max)
min_max([22,4,6,2,23,455,23])
```

[ ]:
```
#
# print(list(range(1,11)))
#  [1x1, 2x2, 3x3, ..., 10x10]
list1 = []
for i in range(1,11):
    list1.append(i*i)
print(list1)

list2 = [x*x for x in range(1,11)]
list2
#
words = ['data','science','machine','learning']
words1 = [len(i) for i in words]
words1
#  words     5
words2 = [i for i in words if len(i) > 5 ]
words2
##
words3 = [j.upper() for i in words for j in i if j in ["a",'e','i','o','u']]
words3
#
def fn(x):
    return x*2
words4 = [fn(a) for a in words]
words4

ls2 =[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
ls3 = [i for i in ls2 if i%2==0]
ls3
ls4 = [i+j for i in "abc" for j in 'abc' if i != j]
ls4
```

[ ]:
```
import os
[n for n in os.listdir(".")]
```

[ ]:
```
d = {'x': 'A', 'y': 'B', 'z': 'C' }
[k + '=' + v for k, v in d.items()]
```

[ ]:
```
# #
#
#       list
```

```
# generator = (x*x for x in range(5))
# for i in generator:
#     print(i)

def fib(max):
    a,b,c = 0,0,1
    while a<max:
        print(c)
        b,c = c,b+c
        a+=1 #a
fib(10)
```

[ ]:
```
#       yield
#     generator    generator    generator
def fib(max):
    a,b,c = 0,0,1
    while a<max:
        yield c
        b,c = c,b+c
        a+=1 #a
    return " "
o = fib(10)
print(next(o))
print(next(o))
print(next(o))
# fib(10)
# #     generator                return
#          generator      next()      yield
#         yield

for n in fib(10):
    print(n)

# for   generator    generator return
#    StopIteration       StopIteration  value

g = fib(6)
while True:
    try:
        x = next(g)
        print(x)
    except StopIteration as e:
        print(e.value)
        break

#yield   next()      Python
# yield       next()         yield
```

[ ]:
```
#
#list  tuple  dict  set  str generator  yield generator function
 ↪Iterable
#  next()            Iterator   generator  yield generator function
from collections.abc import Iterator
isinstance((x for x in range(10)), Iterator)
#   Iterator   list  dict  str   Iterable   Iterator
# list  dict  str  Iterable  Iterator   iter()
isinstance(iter([]),Iterator)
```

[ ]:
```
#
# f = abs
# f(-1)
def add(x,y,f):
    return f(x)+f(y)
add(-5,6,abs)

def f(x):
    return x*x

r = map(lambda x:x*x,[1,2,4,5])
list(r)
list(map(str,[1,2,3,4]))
```

[ ]:
```
#reduce
from functools import reduce
reduce(lambda x,y:x+y,[1,2,3,4])
#filter
list(filter(lambda x:x%2,[1,2,4,45,66]))
list(filter(lambda x:x,[1,0,"",None,66]))
#sorted
sorted(['Bac',"acs","bdc"],key=str.lower) #
sorted([34,-1,4,0,-3],key=abs) #
sorted([34,-1,4,0,-3],key=abs,reverse=True) #
```

[ ]:
```
#
def lazy_sum():
    def sum(*arg):
        total = 0
        for i in arg:
            total+=i
        return total
    return sum

lazy_sum()(1,4,5,7)
```

[ ]:
```
#
def count():
    def f(j):
        def g():
            return j**2

        return g

    fs = []
    for i in range(1,4):
        fs.append(f(i)) #    g
    return fs #
f1,f2,f3 = count()
f1()
f2()
f3()
```

[ ]:
```
#nonloca
#
#
# nonlocal          nonlocal
def inc():
    x = 100
    def fn():
        nonlocal x
        x = x + 1
        return x
    return fn

f = inc()
f()
```

[ ]:
```
#
#    decorator  func
def decorator(func):
    print('    ')
    #        work
    return func

@decorator
def work():
    print('     ')

work()
```

[ ]:
```
def decorator(func):
    print('    ,  000')
```

```
    #       work       work
    def abcd():
        print('   ,  11111')
        func() #      work
        print('   ,  3333')
#       return func()        return
    return abcd

@decorator
def work():
    print('     ,  22222')

work()
```

[ ]:
```
#
def decorator(func):
    #
    def inner(a, b):
        #
        print("    ", a, "+", b)
        func(a, b)
    #
    return inner


#
@decorator
def add_num(a, b):
    print(a + b)

add_num(1,2)
```

[ ]:
```
#

# private   " "     " "     Python
#      private           private
__author__ = 'jack'

def test():
    pass

if __name__ == '__main__':
    test()

# Python       __name__    __main__
#           if         if
#
```

[ ]:
```
#
from datetime import datetime
import threading
import math
i = datetime.now()
week = '    '
print("  :{} {} {}  {}".format(i.year,i.month,i.day,week[i.weekday()]))
print("  :{}:{}:{}".format(i.hour,i.minute,i.second))
#
def run():
    start = datetime.now()
    end = datetime.strptime("2023-07-15 1:23:00","%Y-%m-%d %H:%M:%S")
    duration = end-start
    print("     {} {}:{}:{}".format(duration.days,math.floor(duration.seconds/
 ↪3600),math.floor(duration.seconds%3600/60),duration.seconds%60))
    timer = threading.Timer(1, run)  #
    timer.start()
# run()
```

[2]:
```
# IO
with open(r"1.txt",'w') as f:
    res = f.write("io ")
    print(res)

with open(r"1.txt",'r') as f:
    res = f.read()
    print(res)
```

```
4
io
```

[ ]:
```

```