#!/usr/bin/env python3

# -\*- coding: utf-8 -\*-

"""

Created on Fri Nov 29 10:16:24 2019

@author: hym0509

"""

一，在终端使用交互模式：$python3

1,core data type:

1.1 int

十进制：

10

-5

100

二进制（0b开头，后跟0～1）：

0b0

0b1

0b10

0b11

0b100

0b101

0b110

8进制（0o开头，后跟0～7）：

0o177

0o11

0o10

1.2 float

小数形式：

3.14

3.0

0.14

科学记数：

2.1e8

1.3 complex

1j

1j\*1j

1.4 boolean

True

False

2,运算符

2.1算术运算：

10+5

100-20

2\*3

7/3

7//3

10%3

4\*\*2

2.2 比较运算：

1<2

3>2<4

1+3>2==5

3,print()

print(1+2)

print('hello')

print(1,2)

4,variable

pi=3.14

pi\*10\*\*2

5,statement

5.1 variable name=expression

a=10

b=20

c=a+b

c

a+b

a=a+b

a

5.2 var1,var2=ex1,ex2

a,b=10,20

a+b

c=a

a=b

b=c

a,b=b,a

x,y,z=100,100+100,300

x

y

z

6, is/is not

a=b=100

a is b

a is not b

a=1000

b=1000

a is b

7,id(object)

a=5

id(a)

b=1000

id(b)

c=1000

id(c)

b is c

a=1+4

b=2+3

a is b

id(a)

id(b)

8, multstatement

y+=x == y=y+x

y-=x == y=y-x

y\*=x == y=y\*x

y/x == y=y/x

y=100

y+=1

y

9,数值对象的创建函数

9.1 float(x)

float(3.14)

float("100")

float(True)

9.2 int(x)

int(3.14)

int('123')

9.3 complex(1,2)

9.4 bool(x)

bool(0)

bool(1)

bool(2)

bool(True)

10 数值型函数

10.1 abs(x)

abs(-5)

10.2 round(x,n)

x=123.4567

round(x,1)

round(x,2)

round(x,-1)

10.3 pow(x,y)

pow(3,2)

11 input("")

s=input("请输入字符：")

print("您输入的字符是：",s)

name=input("please enter your name:")

print("hello"+name+"!")

11 字符串运算

11.1 算术运算：+ += \* \*=

x='abcd'+'efg'

x

x='123'

x\*2

x='abcd'

x+='efg'

x

11.2 > >= < <= == !=

'A'<'B'

11.3 in

s='welcome to me'

'to' in s

'yhm' in s

'to' not in s

12, index

s='ABCDE'

s[0]

s[-1]

13,slice

s[1:4]

s[1:2]

s[1:]

s[:2]

s[:]

s[1:4:2]

s[0:0:2]

s[::-1]

s[::-2]

s[4:0:-2]

14,series function

14.1 len(seq)

len(s)

14.2 max(x)

max(s)

14.3 min(x)

min(s)

14.4 ord(c)

ord('A')

print(ord('中'))

14.5 chr(i)

chr(20013)

15. str(obj):

s=123

print(str(s)+'456')

str(True)

str(3.14)

fmt="name:\_%s\_,age:\_%d\_"

name=input('input name:')

age=int(input('input age:'))

s=fmt%(name,age)

print(s)

'name:%s,age:%d'%('yhm',15)

二，statement

1，if statement

if true expression1:

statement1

elif true expression2:

statement2

else:

statementn

s=input("please input a value:")

n=int(s)

if n%2==0:

print('oven')

else:

print('odd')

s=input("please input a value:")

n=int(s)

if 20<n<50:

print(n,"between 20 to 50")

else:

print(n,"outside 20 to 50")

s=input("please input a value:")

n=int(s)

if n==0:

print(n,"0")

elif n>0:

print("positive")

else:

print("negative")

2, if nesting statement

month = int (input("input month:"))

if 1<=month<=12:

print("legal month")

if month<=3:

print("spring")

elif month<=6:

print("summer")

elif month<=9:

print("fall")

else:

print("winter")

else:

print("input wrong")

3, condition expression:expression1 if true expression else expression2

money=int(input("goods total price:"))

pay=money-20 if money>100 else money

print("total expenditure",pay)

s=input("please input a value:")

n=int(s)

absolute=-n if n<0 else n

print("absolute",absolute)

4,pass

score=int(input("input score:"))

if 0<=score<=100:

pass

else:

print("input wrong")

5,bool

5.1 not x

not 0

not 100

5.2 x and y(优先返回假值)

100 and 200

0 and 0.01

200 and 0.0

5.3 x or y（优先返回真值）

100 or 200

100 or 0.0

0 or 200

0 or 0.0

6 循环语句

6.1 statement

while ture expression:

statement1

else:

statement2

#example

i=1

while i<20:

print('hello')

i+=1

else:

print('end')

6.2 recycle nesting

while ture expression1:

while ture expression2:

else:

else:

#example1

j=1

while j<=10:

i=1

while i<=20:

print(i,end='')

i+=1

else:

print()

j+=1

else:

print('end')

6.3 break

#example1

i=1

while i<=6:

print('recycle beginning:',i)

if i==3:

break

print('recycle ending:',i)

i+=1

else:

print('while substatement')

print('finally is:',i)

#example2

import random

n=0

favorite=7

while n<100:

n+=1

draw=random.randint(1,49)

if draw==favorite:

print('got my number!:')

break

else:

print('my favorate did not show up!:c')

print(f'i tried {n} times!')

6.4 dead recycle

#example1

while True:

n=int(input('input:'))

if n==0:

break

print(n)

#example2

y=0

for i in[7,3,4,'x',6,15]:

if not isinstance(i,int):

break

y+=i

print('the total sum is',y)

6.5 for

6.5.1

for variable table in recursive:

statement1

else:

statement2

#example1

s='ABCDE'

for ch in s:

print('ch->',ch)

else:

print('execute else')

print('exit')

#example2

numbers=[7,3,4,5,6,15]

y=0

for i in numbers:

y+=i

print(f'the sum of ''number'' is',y)

6.5.2 range(start,stop,step):

for x in range(4,9,1):

print(x)

6.5.3 for nesting

#example1

for x in 'ABC':

for y in'123':

print(x+y)

#example2

count=0

for x in range(5):

for y in range(10):

count+=1

print(count)

7 continue

7.1 在while中，直接转到while的真值表达式处：

i=0

while i<10:

if i%2==1:

i+=1

continue

print(i)

i+=1

7.2 在for中，从recursive对象中取下一个对象：

#e.g1

for x in range(5):

if x==2:

continue

print(x)

#e.g2

for x in ['a','b','c']:

a=x.upper()

continue

print(x)

print(a)

三 expression

1, sequence

l=[1,2,3,4]

l

1.2,list(iterable):

l=list('abcd')

l

l=list(range(1,10,2))

l

1.3 operation

x=[1,2,3]

y=[4,5,6]

z=x+y

z

x=[1,2,3]

x+=[4,5,6]

x

x=[1,2,3]\*2

x

l=['a',2,'b',3]

print(l[1])

1.4 sequence index assignment

x=[1,2,3,4]

x[2]=3.14

x

1.5 sequence slice

l=list(range(9))

l

l[1:9:2]

l=[2,3.4]

l[3:3]=[4.1,4.2]

l

1.5 sequence slice assignment

l=[2,3,4]

l[0:0]=[0,1]

l

l=[2,3,4]

l[0:1]=[1.1,2.2]

l

l=[2,3,4]

l[1:2]=[3.1,3.2,3.3]

l

l=[2,3,4]

l[:]=[7,8]

l

l[::2]

l[5:]

l[:5]

l=[2,3,4]

l[1:2]=[]

l

lottery=(1,8,9,12,24,28)

len(lottery)

lottery[1:3]

lottery[:4]

lottery[-1]

lottery[-2:]

l=list(range(1,9))

l[1:2]=[2.2,4.4,6.6,8.8]

l

1.6 locate element:l.index(i)

l=[2,3,5,7,11,3]

l.index(5)

l.index(3)

l.index(2)

1.7 l.insert(index,obj)

l.insert(0,1)

l

l.insert(3,4)

l

1.8 l.count(x)

l.count(3)

l.count(2)

l.count(8)

1.9 l.remove(x)

l.remove(1)

l

l.remove(3)

l

1.10 l.append(x)

l=[2,3,5]

l.append(7)

l

l.append([11,13,17,19])

l

1.11 l.clear()

l.clear()

l

1.12 l.extend(list)

l=[2,3,5]

l.extend([7,11,13,1])

l

1.13 l.sort()

l=[4,8,1,3,7]

l.sort()

l

1.14 l.reverse()

l=[8,7,4,3,1]

l.reverse()

l

2. s.split

s='aa bbbb ccccc ddddd'

s.split()

'-'.join(s)

','.join(s)

3 copy

l=[3,1,3,2]

l1=[1,2,l]

l2=l1.copy()

print(l1)

print(l2)

l2[2][0]=3.14

print(l1)

print(l2)

4, 列表推导式

4.1 expression for variable in recursive obj if ture expression

l=[x\*x for x in range(1,10)]

l

l=[x for x in range(1,100) if x%2==1]

l

4.2 nesting:expression1 for variable1 in recursive obj1 if ture expression1

for variable1 in recursive obj1 if ture expression2:

l1=[2,3,5]

l2=[7,11,13]

l3=[x\*y for x in l1

for y in l2]

l3

5, tuple

5.1 tuple function

tuple(range(10))

tuple('abc')

5.2 operation

(1,2,3)+(4,5,6)

(1,2,3)\*3

5.3 t.index(value[,begin[,end]]):获取tuple的位置

t=(2,3,5,7)

t.index(5)

5.4 t.count(value)

t.count(4)

t.count(5)

5.5 reversed(seq)

s='abcd'

l=list(s)

l

l=list(reversed(s))

l

[x for x in reversed(range(10))]

6 dict:{key:value}

d={'name':'tar','age':15}

len(d)

{1:'\_',2:'='}

d={(2008,8,8):'beijin'}

d

6.1 dict() function

d=dict([('name','tar'),('age',15)])

d

d=dict(('ab','cd'))

d

d=dict(('ab','cd',[1,2],(3,4)))

d

dict(name='tar',age=15)

6.2 key index: dict[key]

d=dict(name='tar',age=15)

print(d['age'])

d['name']

6.3 dict[key]=expression

d['name']='tar'

d

{'name':'tar'}

d['age']=15

d

6.4 del dict[key]

del d['age']

d

6.5 in /not in

d={'a':1,'b':2}

'a' in d

1 in d

100 not in d

6.6 recursive

d={'name':'tar',(2002,1,1):'birthday'}

d

for x in d:

print(x)

6.7 推导式：{key expression:value expression for variable in recursive obj [if true expression]}

d={x:x\*\*2 for x in range(10) if x%3==0}

d

7 set

set('abc')

set('abccba')

set({1:'-',2:'=',5:'five'})

d=({1:'-',2:'=',5:'five'})

set(d.values())

set((2,3,5,7))

7.1 operation

7.1.1&交集

s1={1,2,3}

s2={2,3,4}

s1&s2

7.1.2|并集

s1|s2

7.1.3-补集

s1-s2

s2-s1

7.1.4^对称补集

s1^s2

7.1.5<子集

s1<s2

7.1.6>超集

s1>s2

7.1.6in/not in

s={1,'two',3.14}

1 in s

3.14 not in s

s={1,2,3}

for x in s:

print(x)

7.2 comparing

x,y=5,8

print('x<y is',x<y)

print('x>y is',x>y)

print('x==y is', x==y)

print('x!=y is',x!=y)

7.3 chain compasion

x=5

5>=x>4

12<x<20

2<x<10

2<x and x<10

7.4 logical operation

(x==5) and (y==9)

(x==5) or (y==8)

not(x==4) or (y==9)

((x==5)and not(y==8))or(not(x==5)and(y==8))

7.5

7.5.1 s.add()

s.add(100)

s

7.5.2 s.remove()

s.remove(100)

s

7.5.3 s.discard()

s.discard(1)

7.5.4 s.clear()

s.clear()

s

7.5.5 s.copy()

s={1,2,3,4}

s2=s.copy()

s2

7.5.6 s.pop():随机删除一个元素

s.pop()

s

7.6 推导式：{expression for variable in recursive obj [if true expression]}

l=[2,3,5,7,3,5,7,11]

s={x\*\*2 for x in l}

s

7.7 nesting

{x+y for x in 'abc' for y in '123'}

四 function

1,def name(parameters):

statement

1.1 definition

def sayhello():

print('hello')

1.2 alocation

#eg1

def mymax(a,b):

print('a=',a)

print('b=',b)

if a>b:

print(a,'>',b)

else:

print(a,'<=',b)

mymax(20,30)

#eg2

def myfun(a,b):

s=a+b

print('sum',s)

return 1000

v=myfun(20,30)

print('v',v)

1.3 function nesting

def primes(n):

numbers=[2]

def is\_prime(num):

for i in numbers:

if num%i==0:

return False

return True

if n==2:

return numbers

for i in range(3,n+1):

if is\_prime(i):

numbers.append(i)

return numbers

primes(50)

1.4 sequence delivering parameters

def myfun(a,b,c):

pass

s=[1,2,3]

t=myfun(\*s)

print('t',t)

1.5 keywork delivering parameters

def myfun(a,b,c):

print('a is:',a)

print('b is:',b)

print('c is:',c)

myfun(a=11,b=22,c=33)

1.6 dictkey delivering parameters

def myfun(a,b,c):

pass

d={'a':11,'b':22,'c':33}

r=myfun(\*\*d)

print('r',r)

1.7 function missing parameters

def f1(a,b=1,c=2):

print(a,b,c)

f1(100)

f1(100,200)

f1(100,200,300)

2 形参定义方式

2.1 \*形参：

def name(\*元组形参名)：:

statement

#e.g

def func(\*args):

print('number of parameter:',len(args))

func(1,2,3,4)

2.2 keyword形参

def name(\*,keyword形参)：:

statement

#eg

def fn(\*,d,e):

print('d=',d)

print('e=',e)

fn(d=100,e=200)

2.3 \*\*dict形参

def name(\*\* dict形参名):

statement

def func(\*\*kwargs):

print('keyword number:',len(kwargs))

print('kwargs=',kwargs)

func(name='tar',age=15)

2.4 global variable and local variable

a=100

b=200

def fx(c):

d=400

print(a,b,c,d)

fx(300)

print('a=',a)

2.5 globals and locals

a=1

b=2

c=3

def f1(c,d):

e=300

print('locals:',locals())

print('globals:',globals())

f1(100,200)

2.6 inderect function

def f1():

print('hello')

def f2():

print('world')

def fx(fn):

print(fn)

fx(f1)

fx(f2)

2.7 return function

def get\_fx():

s=input('input:')

if s=='max':

return max

elif s=='min':

return min

elif s=='sum':

return sum

l=[2,4,6,8,10]

print(l)

f1=get\_fx()

print(f1(l))

2.8 function nesting

def fn\_outer():

print('fn\_outer is used')

def fn\_inner():

print('fn\_inner is used')

fn\_inner()

print('fn\_outer over')

fn\_outer()

v=100

def fun1():

v=200

print('fun1r=',v)

def fun2():

v=300

print('fun2r=',v)

fun2()

fun1()

print('v=',v)

2.9 global statement: global var1, var2,...

v=100

def fn():

global v

v=200

fn()

print(v)

2.10 nonlocal statement

var=100

def f1():

var=200

print('f1var=',var)

def f2():

nonlocal var

var=300

print('f2var',var)

f2()

print('f2endf2var=',var)

f1()

print('globalvar=',var)

def f1():

v=100

def f2():

v=200

def f3():

nonlocal v

v+=1

f3()

print('f2r=',r)

f2()

print('f2r=',r)

f1()

3 hig order function

3.1 map(func,\*iterables)

def pow2(x):

return x\*\*2

for x in map(pow2,range(1,10)):

print(x)

3.2 filter(func,iterable)

def isodd(x):

return x%2==1

for x in filter(isodd,range(10)):

print(x)

even=[x for x in filter(lambda x:x%2==0,range(10))]

3.3 sorted(iterable,key=none,reverse=false)

l=[5,-2,-4,0,3,1]

l2=sorted(l)

l2

l3=sorted(l,key=abs)

l3

3.4 recursion

def fx(n):

print(n)

if n==3:

return

fx(n+1)

print(n)

fx(1)

五 module

1, import

import math

math.cos(0)

import math as m

length=float(input('length of squire:'))

are=m.pow(length/4,2)

print('area of squire',are)

2, from import

from math import pi

pi\*2\*\*2

from math import sqrt, pow

sqrt(9)

pow(2,3)

3, import sys

def f():

print('fbegin')

import sys

sys.exit(0)

print('fend')

f()

4, import random

import random

x=random.randrange(101)

print(x)

5, generator

5.1 yeild expression

def myyield():

print('genenating 2')

yield 2

print('generating 5')

yield 5

gen=myyield()

it=iter(gen)

next(it)

5.2 expression for variable in recursive [if true expression]

gen=(x\*\*2 for x in range(1,4))

it=iter(gen)

next(it)

next(it)

5.3 zip function

number=[10086,10000,10010,95588]

name=['china move','china tele','china communie']

for n, a in zip(number,name):

print(a,'customer number:',n)

5.4 enumerate(iterable[,start])

name=['china move','china tele','china communie']

for x in enumerate(name):

print(x)

六 OOP

class Dog:

def eat(self,food):

print('dog is eating:',food)

def sleep(self,hour):

print('dog is sleeping:',hour)

def play(self,obj):

print('dog is playing:',obj)

dog1=Dog()

dog1.eat('dog food')

dog1.sleep(1)

dog1.play('ball')

dog2=Dog()

dog2.eat('bond')

dog2.sleep(2)

dog2.play('flight dash')

class Rectangle:

width=0

height=0

def area(self):

return self.width\*self.height

myrectangle=Rectangle()

myrectangle.width=10

myrectangle.height=20

myrectangle.area()

1, attribute

class Dog:

def eat(self,food):

print(self.colour,'s',self.kinds,'is eating',food)

dog1=Dog()

dog1.kinds='chiwawa'

dog1.colour='white'

dog1.eat('bond')

2, \_\_dict\_\_

dog1=Dog()

dog1.\_\_dict\_\_

dog1.kinds='chiwawa'

dog1.\_\_dict\_\_

dog1.colour='white'

dog1.\_\_dict\_\_

3,isinstance(obj,class or tuple)

class Dog:

pass

class Cat:

pass

animal=Dog()

isinstance(animal,Dog)

isinstance(animal,Cat)

4, class inheritance

class Rectangle:

width=0

height=0

def \_\_init\_\_(self,width,height):

self.width=width

self.height=height

def area(self):

return self.width\*self.height

myrectangle=Rectangle(15,30)

myrectangle.area()

class Square(Rectangle):

def \_\_init\_\_(self,length):

super().\_\_init\_\_(length,length)

def diagonal(self):

return(self.width\*\*2+self.height\*\*2)\*\*0.5

mysquare=Square(15)

print(f'area:{mysquare.area()}')

print('diagonal length:',mysquare.diagonal())

5,class variable

class Human:

total\_count=0

def \_\_init\_\_(self,name):

self.name=name

self.\_\_class\_\_.total\_count+=1

print(Human.total\_count)

h=Human('small')

print(h.total\_count)

6,\_\_slots\_\_

class Student:

\_\_slots\_\_=['name','age']

def \_\_init\_\_(self,name,age):

self.name=name

self.age=age

s1=Student('yiming',15)

print(s1.age)

7,\_\_base\_\_

class A:

pass

class B(A):

pass

B.\_\_base\_\_

A.\_\_base\_\_

object.\_\_base\_\_

8, @classmethod

class A:

v=0

@classmethod

def set\_v(cls,value):

cls.v=value

@classmethod

def get\_v(cls):

return cls.v

print(A.get\_v())

A.set\_v(100)

print(A.get\_v)

a=A()

print(A.get\_v())

9, @staticmethod

class A:

@staticmethod

def myadd(a,b):

return a+b

print(A.myadd(100,200))

a=A()

print(a.myadd(300,400))

10 covering method

class A:

def work(self):

print('A.work is used!')

class B:

def work(self):

print('B.work is used!')

b=B()

b.work()

11, super(type,obj):调用其父类的covering method

class A(object):

def work(self):

print('As work usage')

class B(A):

def work(self):

print('Bs work usage')

def super\_work(self):

super().work()

b=B()

b.super\_work

12, hide some detail

12.1 private attribute

class A:

def \_\_init\_\_(self):

self.\_p=10000

def test(self):

print(self.\_p)

a=A()

a.test()

12.2 private method

class B:

def \_m(self):

print('B is used')

def test(self):

self.\_m()

b=B()

b.test()

13.multi-inheritence

class Peasant:

"farmer"

def farm(self,plant):

print('cultivate',plant)

class Worker:

"worker"

def work(self,that):

print('produce',that)

class MigrantWorker(Peasant,Worker):

"farmerworker"

person=MigrantWorker()

person.farm('rice')

person.work('car')

13, reference

13.1 equal but not identical

a = ["Star", "Trek"]

b = ["Star", "Trek"]

c=a

a == b

a == c

a is b

a is c

13.2 copying objects:collecting grades

grades = [1.7, 1.3, 2.7, 2.0]

result = grades.append(1.0)

result

grades

finals = grades

finals.remove(2.7)

finals

grades

14 Side effects

def last\_element(x):

return x.pop(-1)

a = ['Amazon', 'Apple', 'Facebook', 'Google', 'Microsoft', 'Twitter']

last\_element(a)

a

15 Copying

def last\_element(x):

y = x.copy()

return y.pop(-1)

a = ['Amazon', 'Apple', 'Facebook', 'Google', 'Microsoft', 'Twitter']

last\_element(a)

a

16 Deep and shallow copying

fastfood = [["burgers", "hot dogs"], ["pizza", "pasta"]]

italian = fastfood.copy()

italian.pop(0)

american = list(fastfood)

american.pop(1)

american[0] = american[0].copy()

fastfood[0][1] = "chicken wings"

fastfood[1][0] = "risotto"

italian

american