1. 输出两位小数。

```
price = float(input())

GST = price + (price/100)*15

print("Enter a price: %.2f" % price)

print("GST applied: %.2f" % GST)
```

2. 字符串大小写转换

```
str = input("Enter a sentence: ")

str = str.swapcase()

print(str)
```

3. 判断是否为 anagram

```
word1 = input("Enter a word: ")

word2 = input("Enter another word: ")

list1 = list(word1)

list2 = list(word2)

i = 0

list1.sort()

list2.sort()

a = len(word1)


if len(word1) != len(word2):

    T = 0

else:
```

```
    for i in range(0,a):
        if list1[i] != list2[i]:
            T = 0
            break
        else:
            T=1
if T == 1 :
    print(word1,"and",word2,"are anagrams of each other.")
else :
    print(word1,"and",word2,"are not anagrams of each other.")
```

4. 打印范围内被五整除的数，用逗号隔开

```
for i in range(200,295):
    if i%5 == 0 and i%3 != 0 :
        print(i,end = ',')
print(295)
```

5. 列表添加

```
list = []
while True :
    a = int(input("Enter a number: "))
    if a == -999 :
        print(list)
        break
```

```
    else :
        list.append(a)
```

6. 打印 words

```
a = int(input("Enter number of rows: "))
for i in range(1,a+1):
    for j in range(1,i+1):
        print(j,end = ' ')
    if i != a:
        print("")
```

7. 打印星星

```
while True:
    i = 0
    list = []
    star = int(input("Enter number of rows: "))
    for i in range(0,star):
        list.append([])
        for j in range(0,star):
            list[i].append(" ")
    for i in range (0,star):
        if i == 0 or i ==star-1:
            for j in range (star):
                list[i][j]="*"
```

```
            elif i == (star-1)/2:
                list[i][i]="*"
                list[i][0]="*"
                list[i][star-1]='*'
            else :
                list[i][0]="*"
                list[i][star-1]="*"
                list[i][i]="*"
                list[i][star-i-1]="*"
        for i in range (0,star):
            for j in range(0,star):
                print(list[i][j],end="")
            print("")
        break
```

8. 元组

```
a = int(input("Enter an integer: "))
tup =()
for i in range(1,a+1):
    tup1 = (i,)
    tup = tup+tup1
print(tup)
```

9. 字典 ASCII 排序

```
word=input("Enter a word: ")
abc=dict.fromkeys(word)
for i in abc:
    abc[i]= ord(i)
a=sorted(abc.keys())
for i in   a:
    print("{}:{}".format(i,abc[i]))
```

10. 文件中数字的倍数

```
integer = int (input ("Enter an integer: "))
file_name =input ("Enter a filename: ")
f = open (file_name , "r")
neirong = f.readlines()
num=[]
for i in neirong:
    number=i.split()
    for a in number:
        num.append(int(a))
j=0
for i in num :
    if i%number1 ==0:
        j=j+1
if j < 2:
```

```
    print ("There is {} multiple of {} in the \'{}\' file.".format(j,number1,file_name))
else :
    print ("There are {} multiples of {} in the \'{}\' file.".format(j,number1,file_name))
f.close()
```

11. 输出列表前四个元素

```
def getfirst4(items):
    list = []
    for i in range(0,4):
        list.append(items(i))
    return list
```

输出列表中正偶数的和

```
def get_sum_positive_even(numbers):
    a = 0
    for i in numbers:

        if i > 0 and i%2 == 0 :
            a = a + i
    return a
numbers = [1,2,34,2134,3241]
print(get_sum_positive_even(numbers))
```

输出列表中 5 的倍数

```
def get_multiples_of_5(numbers):
```

```
    a = []
    for i in numbers :
        if i > 0 and i%5 == 0:
            a.append(i)
    return a
print(get_multiples_of_5([3, 1, 6, 2, 3, 9, 7, 9, 5, 4, 25]))
```

输出字符串中部分字母的出现次数

```
def count_consonants(word):
    a = 0
    b = list(word)
    consonant = ['A','a','E','e','I','i','O','o','U','u']
    for i in b:
        if i in consonant:
            continue
        else:
            a = a+1
    return a
print(count_consonants('Abracadabra'))
```

拆分字典 按序输出

```
def print_keys_values_inorder(dictionary):
    length=[]
    word=[]
```

```
    for i in dictionary:
        length.append(i)
    length.sort()
    for i in length:
        dictionary[i].sort()
        word.append(dictionary[i])
    k=0
    print(word)
    for i in range(0,len(length)):
        print(length[i],end=" ")
        for j in range(k,len(word)):
            b=word[j]
            for h in b:
                print(h,end=" ")
            k=j+1
            break
        print("")
my_dict = {6:['monday', 'coffee', 'strong'], 5:['short'], 3:['may', 'and']}
print_keys_values_inorder(my_dict)
```

16 translate english into maori words

```
file_name = input("Enter the English to Maori dictionary filename: ")
english = input("Enter an English word: ")
```

```
f = open(filename,'r')
pockage = f.readlines()
17 print the longest word in file
filename = input("Enter a filename: ")
f = open(filename,'r')
pockage = f.readlines()
num = []
for i in pockage :
    a = i.split()
    for j in a :
        num.append(j)
len = len(num)
max = 0
maxname = 0
for i in len :
    if max < len(num[i]) :
        max = len(num[i])
        maxname = num[i]
print("The longest word is",)
```

18 句子中的单词按长度排序

```
a=input("Enter a sentence: ")
sentence = a.split()
```

```
zidian={}
keys=[]
for i in sentence:
    if len(i) not in zidian:
        zidian[len(i)] = i.lower()
    elif i.lower() in zidian[len(i)]:
        pass
    else:
        zidian[len(i)]=zidian[len(i)]+" "+i.lower()
for i in zidian:
    keys.append(i)
keys.sort()
for i in zidian:
    words=zidian[i].split()
    words.sort()
    zidian[i]=" ".join(words)
for i in keys:
    print("{} {}".format(i,zidian[i]))
```

19 删除字符串中的字母

```
def remove_letters(word1, word2):
    result = list(word2)
    for letter in word1:
```

```
            if letter in result:
                result.remove(letter)
    return ''.join(result)
print(remove_letters('hello', 'world'))
```

20 输出文件中的内容

```
def read_content(filename):
    f = open(filename,'r')
    pockage = f.read().splitlines()
    f.close()
    return pockage
```

21 分割字符串，元组输出

```
def get_tag_words(line):
    line = line.split(':')
    tup1 = (line[0],)
    line2 = line[1].split()
    line2.sort()
    tup2 = (line2,)
    tup  = tup1 + tup2
    return tup
```

22 文件内容输出为字典

```
def create_tags_dictionary(filename):
    tags = {}
```

```
    l=[]
    f = open(filename,'r')
    line = f.read().splitlines()
    f.close()
    length = len(line)
    for i in range(0,length):
        l = line[i].split(':')
        line2 = l[1].split()
        line2.sort()
        key = l[0]
        tags[key] = line2
    return tags
tags = create_tags_dictionary('0.txt')
for key in sorted(tags):
    print(key, tags[key])
```

23 字符串拆分为小写单词

```
def get_sorted_unique_words_list(sentence):
    words = sentence.split()
    length = len(words)
    new_words = []
    for i in range(0,length):
        words[i] = words[i].lower()
```

```
    words.sort()
    for i in range(0,length):
        if words[i] not in new_words :
            new_words.append(words[i])
    return new_words
```

24 字典键值交换

```
def get_word_tag_tuple(tags_dictionary, search_word):
    keys = tags_dictionary.keys()
    keys = list(keys)
    for i in keys:
        if search_word in tags_dictionary[i] :
            tup = (search_word,i)
            break
    return tup
dict = {'NN': ['dreamer', 'father', 'fun', 'grass', 'mother', 'odense', 'rain', 'shoemaker',
'spring', 'summer', 'tortoise', 'toy', 'washerwoman'],'IN':['abc']}
print(get_word_tag_tuple(dict, 'abc'))
```

25 访问元组并写入字典

```
def get_tags_frequency(list_of_tuples) :
    dictionary = {}
    length = len(list_of_tuples)
    for i in range(0,length):
```

```
        if list_of_tuples[i][1] not in dictionary :
            dictionary[list_of_tuples[i][1]] = 1
        else :
            num = dictionary[list_of_tuples[i][1]]
            num += 1
            dictionary[list_of_tuples[i][1]] = num
    return dictionary

list_of_tuples = [('a', 'DT'), ('and', 'CC'), ('father', 'NN'), ('his', 'PRP$'), ('mother', 'NN'),
('shoemaker', 'NN'), ('was', 'VBD'), ('washerwoman', 'NN')]
freq_dict = get_tags_frequency(list_of_tuples)
for key in sorted(freq_dict.keys()):
    print(key, freq_dict[key])
```

26 处理并输出字典内容

```
def print_dictionary(tags_dictionary) :
    list1 = tags_dictionary.keys()
    list1 = list(list1)
    list1.sort()
    for i in list1 :
        print(i,tags_dictionary[i])
    return 0
```

```
tags_dictionary = {'DT': 1, 'CC': 1, 'NN': 4, 'PRP$': 1, 'VBD': 1}
print_dictionary(tags_dictionary)
```

27 处理字典并输出内容

```
def print_all_phrases(tags_dictionary) :
    list1 = tags_dictionary['DT']
    list2 = tags_dictionary['JJ']
    list3 = tags_dictionary['NN']
    for i in list1:
        for j in list2:
            for k in list3:
                print(i,j,k)
    return 0

tags = {'DT': ['a','one'], 'NN': ['father', 'mother', 'room', 'shoemaker', 'washerwoman'],
'JJ': ['poor','rich']}
print_all_phrases(tags)
```

27 处理字典并输出随机某条内容

```
import random
def print_random_phrase(tags_dictionary) :
    list1 = tags_dictionary['DT']
    list2 = tags_dictionary['JJ']
    list3 = tags_dictionary['NN']
```

```
    list4 = []
    for i in list1:
        for j in list2:
            for k in list3:
                list5 = [i,j,k]
                list4.append(list5)
    length = len(list4)
    m = random.randrange(0,length)
    '''
    len1 = len(list1)
    len2 = len(list2)
    len3 = len(list3)
    i   = random.randrange(0,len1)
    j = random.randrange(0,len2)
    k = random.randrange(0,len3)
    '''
    print(list4[m][0],list4[m][1],list4[m][2])
    return 0

tags = {'JJ': ['brown', 'yellow'], 'NN': ['grass', 'summer'], 'DT': ['the', 'a']}
print_random_phrase(tags)
```

29 报错 1-number

```
def is_valid_score(score):
    try :
        if score >= 0 and score <= 100 :
            return 'True'
        else :
            raise ValueError()
    except TypeError:
        return "ERROR: Invalid score!"
    except ValueError:
        return "ERROR: Invalid score!"
```

30 报错 2-number

```
def is_valid_radius(radius):
    try :
        if radius > 0   :
            return 'True'
        else :
            raise ValueError()
    except TypeError:
        return "ERROR: Invalid radius!"
    except ValueError:
        return "ERROR: Invalid radius!"
print(is_valid_radius(16))
```

```
print(is_valid_radius(-1))

print(is_valid_radius('12'))

print(is_valid_radius([16,12]))

print(is_valid_radius(2.5))

print(is_valid_radius(0))
```

31 报错 3-str

```
def count_consonants(word):
    try :
        n = 0
        list1 =
['b','c','d','f','g','h','j','k','l','m','n','p','q','r','s','t','v','w','x','y','z','B','C','D','F','G','H','J','K','L
','M','N','P','Q','R','S','T','V','W','X','Y','Z']
        if type(word) != str :
            raise TypeError ()
        for i in word :
            if i in list1 :
                n += 1
        return n
    except TypeError:
        return "ERROR: Invalid input!"
```

32 报错 4-list

```
def set_list_element(a_list, index, value):
```

```
    try :
        a_list[index] = value
    except TypeError :
        return print('ERROR: Invalid input.')
    except IndexError :
        return print('ERROR: Invalid index: {}.'.format(index))
```

33 报错 5-列表最大值

```
def get_max(numbers):
    try :
        length = len(numbers)
        max = -1000000000
        for i in range(0,length):
            if max < numbers[i] :
                max = numbers[i]
        return float(max)
    except TypeError :
        return 'ERROR: Invalid number!'
```

34 报错 6-列表偶数相加

```
def check(number):
    try :
        if number % 2 == 0 :
            return int(number)
```

```
        else :
            return int(0)
    except TypeError :
        return int(0)


def get_sum_even(numbers):
    length = len(numbers)
    num = 0
    i = 0
    for i in range(0,length):
        num = num + check(numbers[i])


    return num
```

35 报错 7-求体积

```
def get_volume(radius, height):
    try :
        if radius < 0 and height < 0 :
            raise HRError()
        elif radius < 0 :
            return 'ERROR: Radius must be positive.'
        elif height < 0 :
            return 'ERROR: Height must be positive.'
```

```
        elif height == 0 or radius == 0 :
            return 'ERROR: Not a cylinder.'
        else :
            num = 3.1415926535 * radius * radius * height
            return int(num+0.5)
    except TypeError :
        return 'ERROR: Invalid input.'
    except HRError :
        return 'ERROR: Height and radius must be positive.'
```

36 报错 8-字典

```
def get_maori_word(dictionary, word):
    try :
        maori_word = dictionary[word]
        return maori_word
    except KeyError :
        return "ERROR: {} is not available.".format(word)
```

```
dictionary ={'example': 'tauira', 'house': 'whare', 'apple': 'aporo', 'love': 'aroha', 'food':
'kai',
```

'hello': 'kiaora', 'work': 'mana', 'weather': 'huarere', 'greenstone': 'pounamu',

'red': 'whero', 'orange': 'karaka', 'black': 'mangu'}

37 报错 9-字典 2

```
def get_phone(phones_dictionary, name):
    try :
        if name == "":
            raise ValueError()
        if type(name) != str :
            raise TypeError()
        number = phones_dictionary[name]
        return number
    except KeyError :
        return "ERROR: {} is not available.".format(name)
    except ValueError:
        return "ERROR: Invalid name!"
    except TypeError:
        return "ERROR: Invalid input!"
```

```
phones_dictionary = {'Martin':8202, 'Angela':6620, 'Ann':4947, 'Damir':2391,
'Adriana':7113, 'Andrew':5654}
```

38 报错 10-处理文件并输出内容

```
def read_scores(filename):
    try:
        if type(filename) != str :
            raise TypeError()
        if filename == '':
            raise NameError()
        input_file = open(filename, "r")
        scores = input_file.read().split()



        new_list = scores
        length = len(new_list)
        if length == 0:
            raise ZeroDivisionError()
        check = 0
        for i in range(0,length):
            if float(new_list[i]) > 0 :
                check += 1
        if check == 0 :
```

```
            raise ZeroDivisionError()



        numbers = [float(score) for score in scores if float(score) >= 0 ]
        input_file.close()
        number_of_marks = len(numbers)
        total_marks = sum(numbers)
        print("There are {} score(s).".format(number_of_marks))
        print("The total is {:.2f}.".format(total_marks))
        print("The average is {:.2f}.".format(total_marks/number_of_marks))


except TypeError :
        return print("ERROR: Invalid input!")
except FileNotFoundError:
        return print("ERROR: The file \'{}\' does not exist.".format(filename))
except NameError:
        return print("ERROR: Invalid filename!")
except ZeroDivisionError :
        return print("ERROR: No positive scores in the input file.")
except OSError:
        return print("ERROR: Invalid input!")
except ValueError:
```

```
return print("ERROR: The input file contains invalid values.")
```