COMP 1001 Rocky K. C. Chang

Part 1:

0.0 **Introduction:**

None

0.0 **CS education:**

None

1.0 **Computers and programs:**

None

1.1 **Computer numbers:**

Decimal → Binary*

1.2 **Primers on iterations and functions**

What is function and what is iteration (or looping informally).

Loop invariance（to prove a function is correct）

1.3 **Pseudocoding**

Function: ###

Input: ###

Output: ###

- for each … in…do…:
- x ← y; x ← function ()
- if…do…:
- while…do…:
- return …

2.1 **Computation**

PROC0, 1, 2, 3

MCGW; Timetabling; Hanoi

3. **MCGW problem**

The procedure in this problem is crucial.

4. **Stock price problem**

This problem is really important and will **definitely** appear in the Exam.

Important!

1.0 **Starting using python**

None

2.0 **Basic data types**

ord (A) = 65; chr (65) = A

转义字符	描述
\(在行尾时)	续行符
\\	反斜杠符号
\'	单引号
\"	双引号

\a	响铃
\b	退格(Backspace)
\e	转义
\000	空
\n	换行
\v	纵向制表符
\t	横向制表符
\r	回车
\f	换页
\oyy	八进制数 yy 代表的字符，例如：\o12 代表换行
\xyy	十进制数 yy 代表的字符，例如：\x0a 代表换行
\other	其它的字符以普通格式输出

round(x, n) built-in function→ 四舍五入到 n 位
math.ceil(x) math function → 返回数字的上入整数
math.floor(x) math function → 返回数字的下舍整数

x, y = eval(input("Input the first and second numbers separated by a comma: "))
用 int 在这里是不行的

Operation	Result
`x + y`	sum of *x* and *y*
`x - y`	difference of *x* and *y*
`x * y`	product of *x* and *y*
`x / y`	quotient of *x* and *y*
`x // y`	floored quotient of *x* and *y*
`x % y`	remainder of `x / y`
`-x`	*x* negated
`+x`	*x* unchanged
`abs(x)`	absolute value or magnitude of *x*
`int(x)`	*x* converted to integer
`float(x)`	*x* converted to floating point
`complex(re, im)`	a complex number with real part *re*, imaginary part *im*. *im* defaults to zero.
`c.conjugate()`	conjugate of the complex number *c*
`divmod(x, y)`	the pair `(x // y, x % y)`
`pow(x, y)`	*x* to the power *y*
`x ** y`	*x* to the power *y*

3. **Decision structures**
   if & while → loops

not > and > or

4. **Sequences→ List. String. Tuple**
   list [1,2,3,4] & tuple (1,2,3,4)

Operation	fruit = ['banana', 'apple, 'cherry']	
Replace	fruit[2] = 'coconut'	['banana', 'apple', 'coconut']
Delete	del fruit[1]	['banana', 'cherry']
Insert	fruit.insert(2, 'pear')	['banana', 'apple', 'pear', 'cherry']
Append	fruit.append('peach')	['banana', 'apple', 'cherry', 'peach']
Sort	fruit.sort()	['apple', 'banana', 'cherry']
Reverse	fruit.reverse()	['cherry', 'banana', 'apple']

Remove 会移除第一个匹配项
Pop 会默认移除最后一项,并返回这个值
Sequence:

Operation		String s = 'hello' w = '!'	Tuple s = (1,2,3,4) w = (5,6)	List s = [1,2,3,4] w = [5,6]
Length	len(s)	5	4	4
Select	s[0]	'h'	1	1
Slice	s[1:4] s[1:]	'ell' 'ello'	(2, 3, 4) (2, 3 ,4)	[2 ,3, 4] [2, 3, 4]
Count	s.count('e') s.count(4)	1 *error*	0 1	0 1
Index	s.index('e') s.index(3)	1 --	-- 2	-- 2
Membership	'h' in s	True	False	False
Concatenation	s + w	'hello!'	(1, 2, 3, 4, 5, 6)	[1, 2, 3, 4, 5, 6]
Minimum Value	min(s)	'e'	1	1
Maximum Value	max(s)	'o'	4	4
Sum	sum(s)	*error*	10	10

range(*start*, *stop*[, *step*])
list(range(10, 0, -1))
→[10, 9, 8, 7, 6, 5, 4, 3, 2, 1]

end 的妙用：
默认 end = ‘/n’ end = ‘’ 里面可以是分隔符。

5.0 **Sequences 2 → Complex!**

- str.capitalize()
- str.casefold()
- str.center(width[, fillchar])
- str.count(sub[, start[, end]])
- str.title()
- str.translate(map)
- str.upper()
- str.zfill(width)
- student.split()

student.split(),
student.lower(),
student.upper(),
student.capitalize().

Operation	Result	Notes
`x in s`	`True` if an item of *s* is equal to *x*, else `False`	(1)
`x not in s`	`False` if an item of *s* is equal to *x*, else `True`	(1)
`s + t`	the concatenation of *s* and *t*	(6)(7)
`s * n` or `n * s`	equivalent to adding *s* to itself *n* times	(2)(7)
`s[i]`	*i*th item of *s*, origin 0	(3)
`s[i:j]`	slice of *s* from *i* to *j*	(3)(4)
`s[i:j:k]`	slice of *s* from *i* to *j* with step *k*	(3)(5)
`len(s)`	length of *s*	
`min(s)`	smallest item of *s*	
`max(s)`	largest item of *s*	
`s.index(x[, i[, j]])`	index of the first occurrence of *x* in *s* (at or after index *i* and before index *j*)	(8)
`s.count(x)`	total number of occurrences of *x* in *s*	

https://docs.python.org/3/library/stdtypes.html

list’ s Function:
append ()
join ()

format

3.1415926	{:.2f}	3.14	保留小数点后两位
3.1415926	{:+.2f}	+3.14	带符号保留小数点后两位
-1	{:+.2f}	-1.00	带符号保留小数点后两位
2.71828	{:.0f}	3	不带小数
5	{:0>2d}	05	数字补零 (填充左边，宽度为 2)
5	{:x<4d}	5xxx	数字补 x (填充右边，宽度为 4)
10	{:x<4d}	10xx	数字补 x (填充右边，宽度为 4)
1000000	{:,}	1,000,000	以逗号分隔的数字格式
0.25	{:.2%}	25.00%	百分比格式
1000000000	{:.2e}	1.00e+09	指数记法
13	{:10d}	13	右对齐 (默认，宽度为 10)
13	{:<10d}	13	左对齐 (宽度为 10)
13	{:^10d}	13	中间对齐 (宽度为 10)

6.0 **Dictionaries and sets**

Operation	Results
`dict()`	Creates a new, empty dictionary

`dict(s)`	Creates a new dictionary with key values and their associated values from sequence `s`, for example,
	`fruit_prices = dict(fruit_data)`
	where `fruit_data` is (possibly read from a file):
	`[['apples', .66],…,['bananas', .49]]`

`len(d)`	Length (num of key/value pairs) of dictionary `d`.
`d[key] = value`	Sets the associated value for `key` to `value`, used to either add a new key/value pair, or replace the value of an existing key/value pair.
`del d[key]`	Remove key and associated value from dictionary `d`.
`key in d`	`True` if key value `key` exists in dictionary `d`, otherwise returns `False`.

- items (): all the key-values pairs as a list of tuples
- keys (): all the keys as a list
- values (): all the values as a list
- dict.update (dict1): add dict1 into dict
- set.add()
- set.remove()

Set operator	Set A = {1,2,3}	Set B = {3,4,5,6}	
membership	`1 in A`	True	*True if 1 is a member of set*
add	`A.add(4)`	{1,2,3,4}	*Adds new member to set*
remove	`A.remove(2)`	{1,3}	*Removes member from set*
union	`A | B`	{1,2,3,4,5,6}	*Set of elements in either set A or set B*
intersection	`A & B`	{3}	*Set of elements in both set A and set B*
difference	`A - B`	{1,2}	*Set of elements in set A, but not set B*
symmetric difference	`A ^ B`	{1,2,4,5,6}	*Set of elements in set A or set B, but not both*
size	`len(A)`	3	*Number of elements in set (general sequence operation)*

7.0 Functions

mutable immutable

8.0 Files

```
infile = open("a_text_file_of_your_choice","r")
data = infile.read()
print(data)
```