Chapter 11 Mind map

Algorithm
- What?
  - A sequence of steps that can be carried out to perform a task.
  - the recipe
- How? (Expressing)
  - Concise, elegant and easy to understand
  - Structured English: a subset of the English language that consists of command statements used to describe an algorithm
  - Flowchart: shapes linked together to represent the sequential steps of an algorithm
    - Input
    - Assignment
    - Sequence
    - Selection
    - Repetition
    - Output
  - Pseudocode: a way of using keywords and identifiers to describe an algorithm without following the syntax of a particular programming language.

Constructs for computing algorithms

Page 129

Array
- Linear Search: Checking each element of an array in turn for a required value
- Bubble Sort: a sort method where adjacent pairs of values are compared and swapped

process

1D: linear list
2D: table / matrix
ThisTable[row, column] ← value
2D → use nested loops

Variable
- a storage location for a data value that has an identifier
- not contain spaces, only letters, digits and _
- naming of it should reflect the use → often more than 1 word
  - ↳ camelCaps.

Assignments
- Assigning a value: INPUT A ; A ← 1
  - Copying a value: B ← A
- Updating a value: A ← A + 1
  - Swapping 2 values: Temp ← A, A ← B, B ← Temp

Identifier table: a table listing the variable identifiers required for the solution, with explanations

Logic Statements
- =, <, >, <=, >=, <>
- IF <Condition> THEN <statement(s)> ELSE <statement(s)> ENDIF
  - Nested IF Statements: conditional statement within conditional statements

Loops
- REPEAT ... UNTIL <condition>
- FOR ... ... ENDFOR
- WHILE <condition> ... ENDWHILE
  - Rogue value: a value used to terminate a sequence of values
  - Nested Loops: loop containing another loop

Definition

Shirley Zeng 曾雯 S2C1

Chapter 12 Mind Map

breaking down the steps of an outline solution into smaller and smaller steps

Stepwise refinement

Decompose the problem into sub-tasks

What?

procedures & functions

Modules

a sequence of steps that is given an identifier and can be called to perform a sub-task

a sequence of steps that is given an identifier and returns a single value, function call is part of an expression

local

a variable that is only accessible within the module where it is declared

variable

global

a variable that is accessible from all modules

Structure charts

Structure Charts:
a graphical representation of the modular structure of a solution

Parameter:
a value passed between modules

downwards pointing arrow: a parameter supplying a value to a lower-level module

upwards pointing arrow: a parameter supplying a new value to the module at the next higher level

selection:
diamond shape → TRUE or FALSE

repetition:
semi-circular arrow → repetition of the modules below the arrow

an arrow with a solid roundend → the value transferred is a flag (a Boolean value)

a double-headed arrow → the variable value is updated within the module

Pg 160-167 ☆☆☆
understand

m definition
Shirley Zeng 曾昱.

No.

Date

Chapter 13 Mind Map (?)

1. Key Characteristics of Python

- Every statement must be on a seperate line
- off-side rule: indentation is significant
- keywords are written in lower case
- case-sensitive: Number1 ≠ number1
- everything is an object
- code makes extensive use of a concept called "slicing"
- programmes are interpreted

2. ① declaration of variables

in Python, does not have, instead, tags values

② declaration and assignment of constants

CONSTANT <identifier> = <value>

<identifier> = <value> PI = 3.14 (capital letters)

③ Assignments of variables

<identifier> ← <expression>

<identifier> = <expression> A=34; B=B+1

④ Arithmetic operators

Addition, Subtraction, Multiplication, Division

\+ +, - -, * *, / /

Exponent, Integer division, Modules

^ **, DIV //, MOD %

Rules of precedence: Parentheses, exponentiation, multiplication, division, addition, subtraction

define the order of the calculations to be performed

⑤ Outputting information to the screen

OUTPUT <string> OUTPUT <identifier(s)> OUTPUT "string" // no newline

print(<printlist>) print(<printlist>, end="") print("Hello {0}. Your name is {1}".format(YourName, Number1))

seperate by commas ← avoid moving onto the next line

⑥ Getting input from the user

INPUT "Prompt: " A

could be converted into other type ← A = input("Prompt: ") ① single could be accepted

type(A) = str ↓ parameter to the input-function

⑦ Comments

this is a comment

3. Data types

INTEGER	int	
REAL	float	
CHAR	not available	
STRING	str	'', "", """" stored as ASCII but Unicode strings are also available
BOOLEAN	bool	TRUE = 1, FALSE = 0
DATE	not available	as a built-in
CURRENCY	not available	

4. Boolean expressions

==, <> !=, <, >, >=, <=

AND and, OR or, NOT not

No.

Date

5. Selection

IF <Boolean expression>
THEN ①
② <statement(s)>
ELSE
<statement(s)>
ENDIF ③

IF...THEN...ELSE

if <Boolean expression>: ①
② <statement(s)>
else:
<statement(s)>

Nested IF

IF <Boolean expression>
THEN
<statement(s)>
ELSE ①
IF <Boolean expression>
THEN
<statement(s)>
ELSE
<statement(s)>
ENDIF
ENDIF

if <Bex>:
<s(s)>
elif <Bex>: ①
<s(s)>
else:
<s(s)>

Case

does not have → use Nested IF

CASE OF <expression>
<value 1> : <statement(s)>
<value 2>, <value 3> : <statement(s)>
<value 4> TO <value 5>: <statement(s)>
OTHERWISE <statement(s)>
ENDCASE

6. Iteration

Know how many times → count-controlled loop

Depends on some condition determined by what happens within the loop → conditional loop

count-controlled (for) loops

FOR <control variable> ← s TO e STEP i // STEP is optional
<statement(s)>
ENDFOR

default: s = 0, i = 1

for <control variable> in range(s,e,i): ①
<statement(s)>

integer

finish below

Post-condition loops ~~do~~ not available → use pre-condition loop

Pre-condition loops

WHILE <condition>
<statement(s)>
ENDWHILE

while <condition>: ①
<statement(s)>

indented less (intended by a set number of spaces)

No.

Date

7. Array

in Python, does not have. Equivalent data structure → List (1D), lists of lists (2D)

list: ordered sequence of items that don't have to be of the same data structure
↓ dynamic

① Accessing 1D arrays

NList[25]=0 NList[24]=0

② Creating 2D arrays

Board[1:6, 1:7] OF INTEGER

Board = [[0 for i in range(7)] for j in range(6)]
Board = [[0]*7]*6

③ Accessing 2D arrays

Board[3,4]=0 ← row ← column Board[2][3]=0

8. Built-in functions

- string manipulation functions Py #202
- slicing ThisString = "ABCDEFG" ThisString[2:] = "CDEFG"
  [:2] = AB
  [-2:] = FG
  [:-2] = ABCDE
- rounding numbers round(xI, ndigits)
  round(3.1415, 2) = 3.14
  round(3.1415) = 3
- truncating numbers int(x) int(3.1) = 3 int(3.8) = 3
- converting a string to a number int(s), float(x)
- formatting numbers for output → tabulated way
  ^ centres the value < align the value on the left
  w overall character width > align the value on the right
  xf x → number of decimal places for floating point number
- random number generator randint(1, 6) → between 1 & 6 inclusive
- date and time functions

from datetime import * import the library

2015-03-15 ← print SomeDate = date(2015,3,15) year, month, day

2014-08-29 ← Today = date.today() read system clock

2015-03-16 ← print SomeDate = SomeDate + timedelta(1) move on one day

KOKUYO

No.

Date

9. Text files

writing

```
OPENFILE <filename> FOR WRITE
WRITEFILE <filename>, <stringValue>
CLOSEFILE
```

```
FileHandle = open("SampleFile.TXT", "w")
FileHandle.write(LineOfText)
FileHandle.close()
```

"\n" → move to the next line

write.

sth you write

reading

```
OPENFILE <filename> FOR READ
READFILE <filename>, <stringVariable>
CLOSEFILE
```

```
FileHandle = open("SampleFile.TXT", "r")
LineOfText = FileHandle.readline()
FileHandle.close
```

read

which line

appending

```
OPENFILE <filename> FOR APPEND
WRITEFILE <filename>, <stringValue>
CLOSEFILE
```

```
FileHandle = open("SampleFile.TXT", "a")
FileHandle.write(LineOfText)
FileHandle.close()
```

append

The end-of-file (EOF) marker

```
OPENFILE "Test.txt" FOR READ
WHILE NOT EOF("Test.txt")
    READFILE "Test.txt", TextString
    OUTPUT TextString
ENDWHILE
CLOSEFILE "Test.txt"
```

```
FileHandle = open("Test.txt", "r")
LineOfText = FileHandle.readline()
while len(LineOfText) > 0:
    LineOfText = FileHandle.readline()
    print(LineOfText)
FileHandle.close()
```

control.

Campus