香港中文大學（深圳）
The Chinese University of Hong Kong, Shenzhen

Introduction to Computer Science: Programming Methodology

Midterm Review

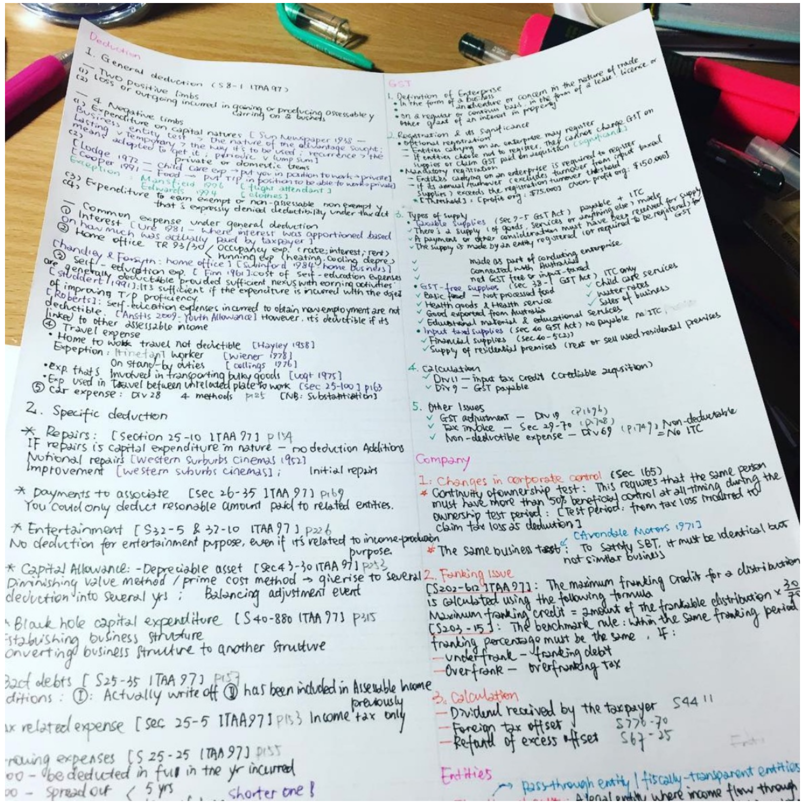
Prof. Pinjia He
School of Data Science

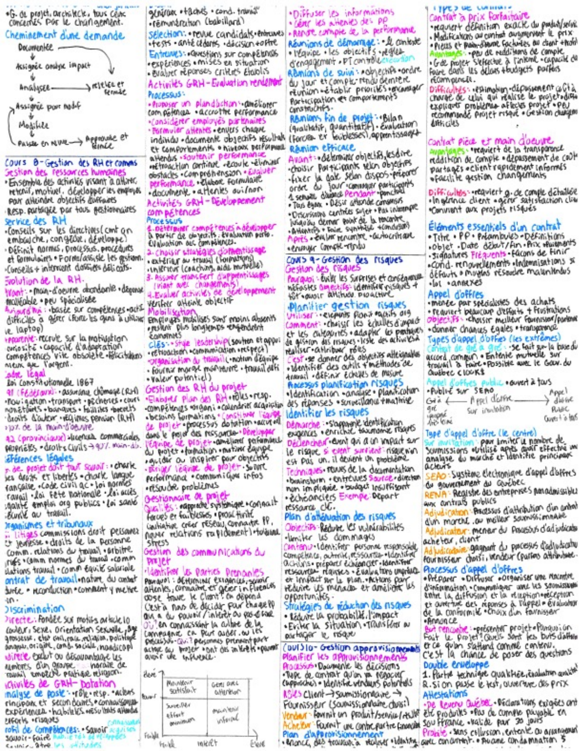
Scope

- Lecture **1** – Lecture **5** (List)
- Question: MC, Short-answer questions

Cheat sheet

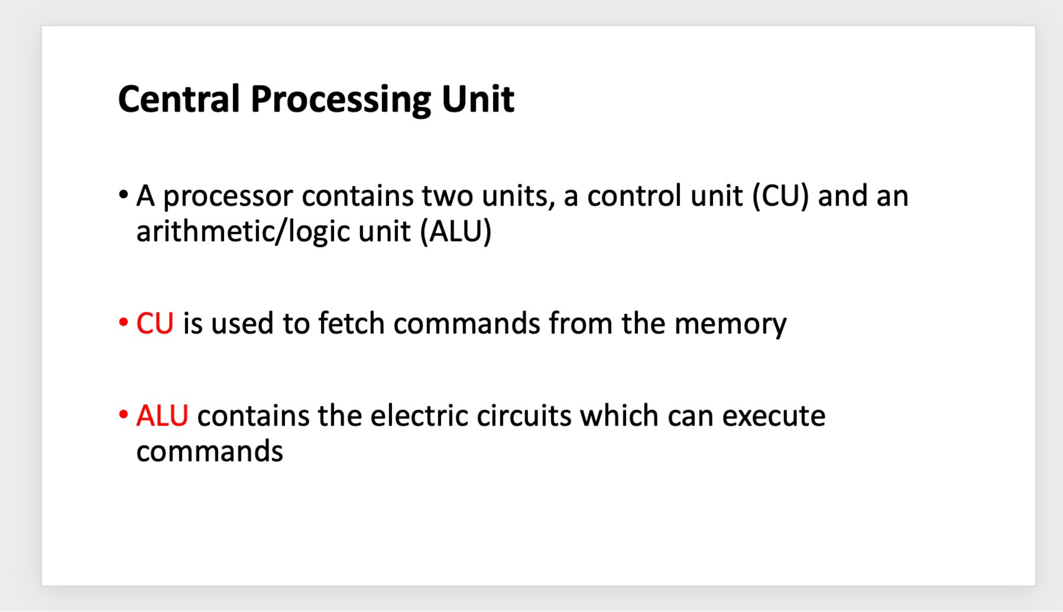
- A4 paper **both sides**
- Hand-written or printed





Tips about cheat sheet

- **Concepts** on the cheat sheet

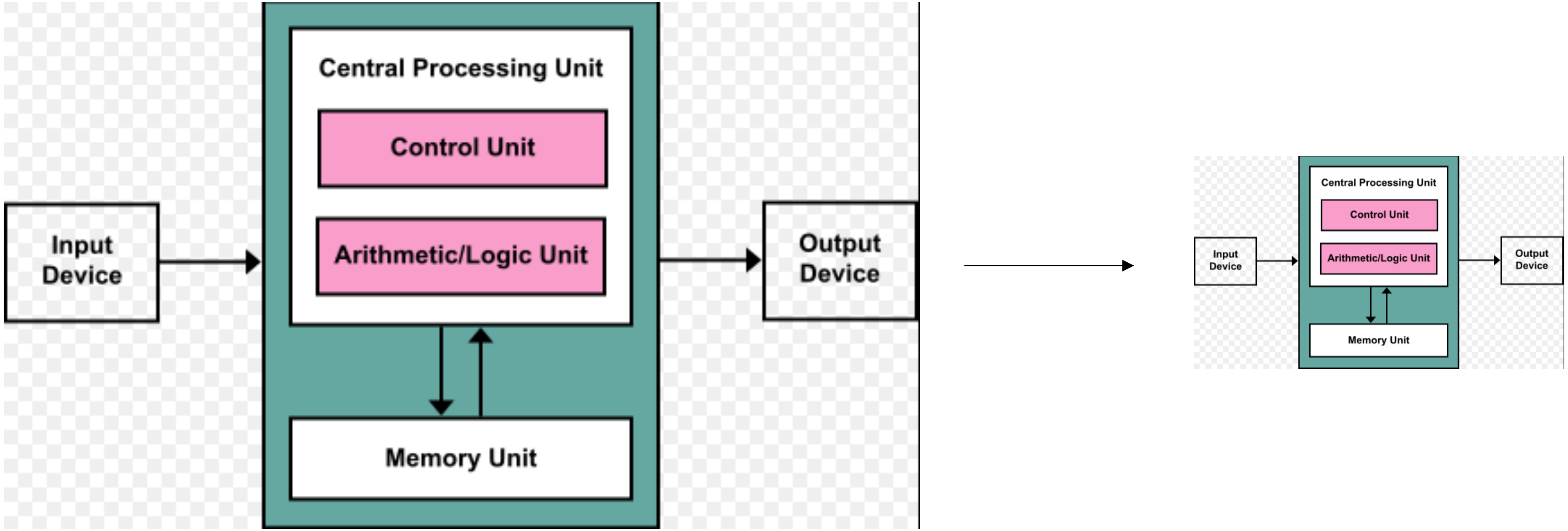


Central Processing Unit

- A processor contains two units, a control unit (CU) and an arithmetic/logic unit (ALU)
- CU is used to fetch commands from the memory
- ALU contains the electric circuits which can execute commands

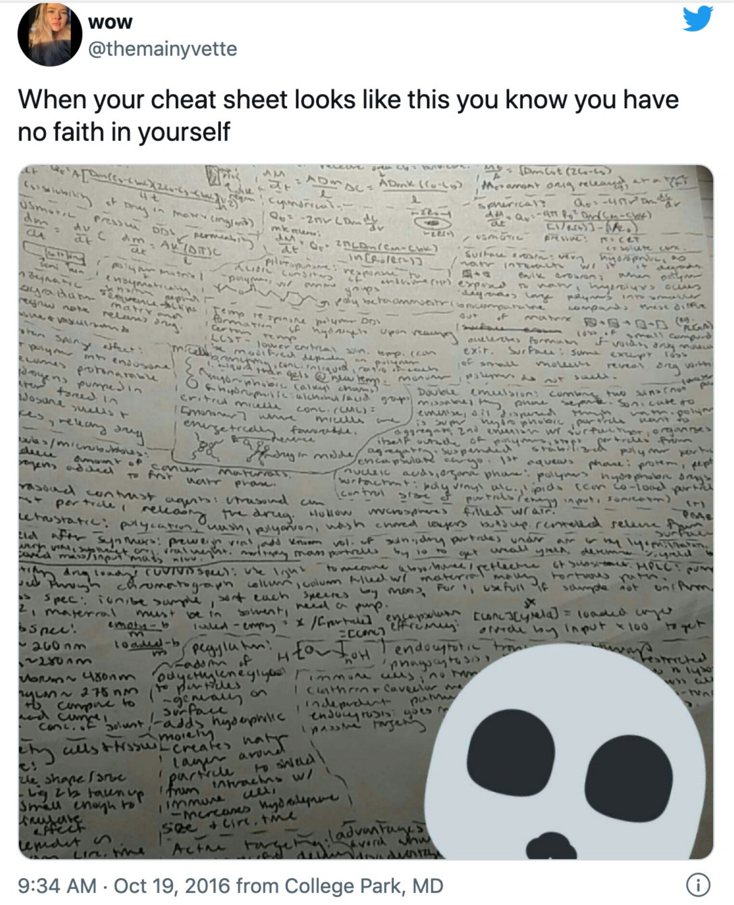
Tips about cheat sheet

- **Resize** the images/screenshots



Tips about cheat sheet

- Make sure you **can find** the right place



wow
@themainyvette

When your cheat sheet looks like this you know you have no faith in yourself

9:34 AM · Oct 19, 2016 from College Park, MD

Details!

“Hello world” v.s. “ Hello world”

1+1 v.s. ‘1’+’1’

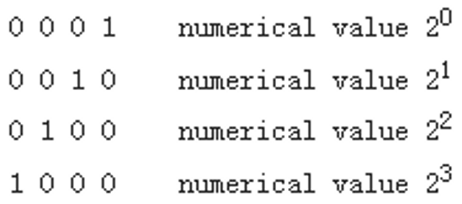
```
a = ‘113’
a[1] = ‘2’
print(a)
```

Everything from slides

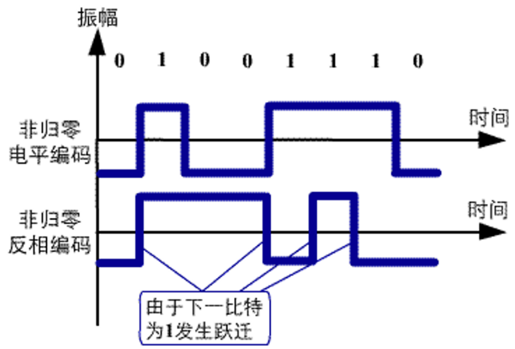
What can a computer actually understand?

- The computers used nowadays can understand only binary number (i.e. 0 and 1)
- Computers use voltage levels to represent 0 and 1
- NRZL and NRZI coding
- The instructions expressed in binary code is called machine language

(Video: Programming Languages)



0 0 0 1	numerical value 2^0
0 0 1 0	numerical value 2^1
0 1 0 0	numerical value 2^2
1 0 0 0	numerical value 2^3

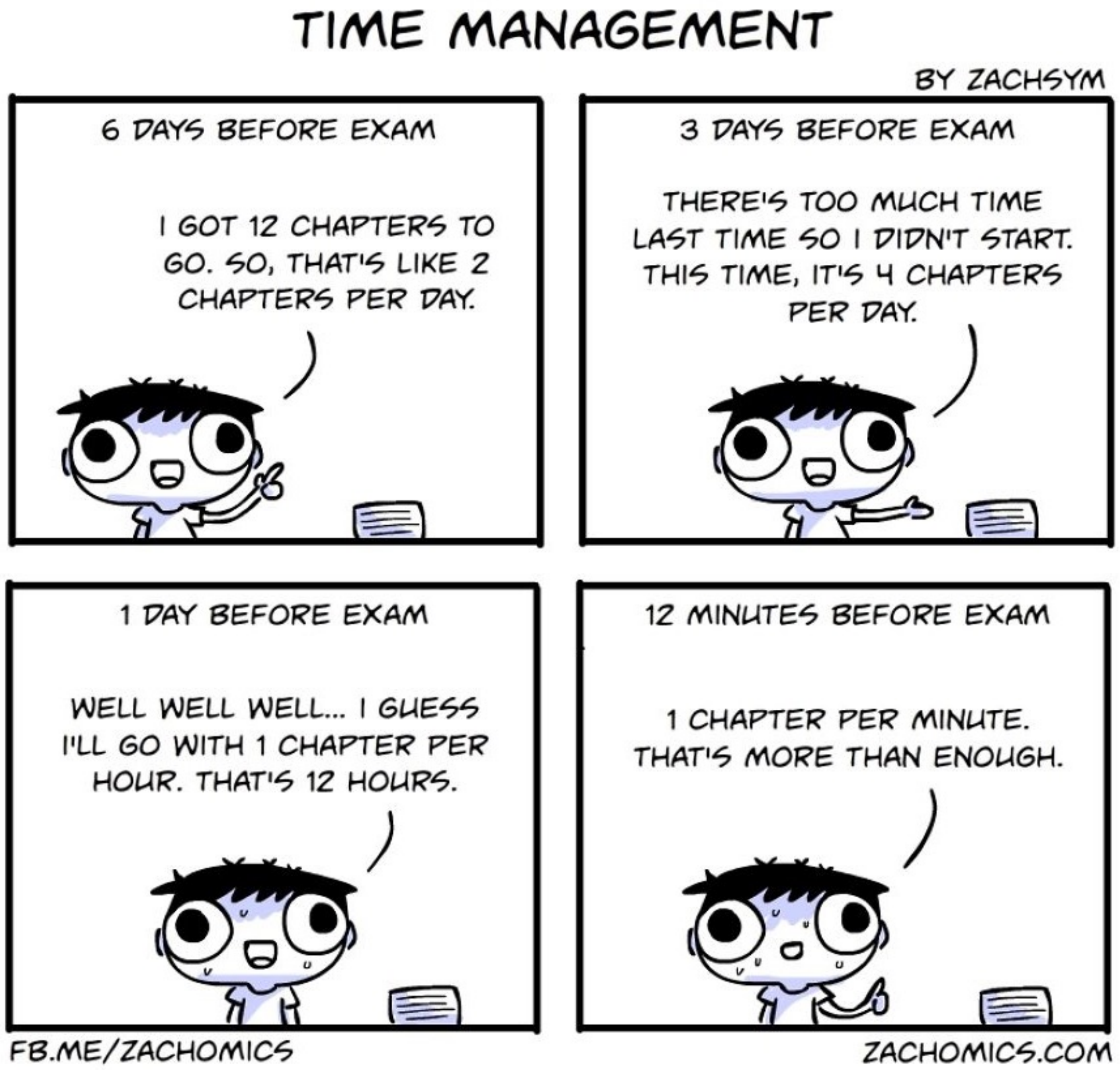


Other tips

- Carefully read the question, whether it ask you to select **ONE** or allows **MULTIPLE**
- Write down what you know
- **Example**(s) on your cheat sheet

Other tips

- Practice makes perfect :)
- Start **early**


TIME MANAGEMENT
BY ZACHSYM
6 DAYS BEFORE EXAM
I GOT 12 CHAPTERS TO GO. SO, THAT'S LIKE 2 CHAPTERS PER DAY.
3 DAYS BEFORE EXAM
THERE'S TOO MUCH TIME LAST TIME SO I DIDN'T START. THIS TIME, IT'S 4 CHAPTERS PER DAY.
1 DAY BEFORE EXAM
WELL WELL WELL... I GUESS I'LL GO WITH 1 CHAPTER PER HOUR. THAT'S 12 HOURS.
12 MINUTES BEFORE EXAM
1 CHAPTER PER MINUTE. THAT'S MORE THAN ENOUGH.
FB.ME/ZACHOMICS
ZACHOMICS.COM

MC

Concerning Von Neumann Architecture, which is the following is incorrect?

A. It has central processing unit

B. It has memory unit

C. It has HDMI cable

D. It has input device

MC

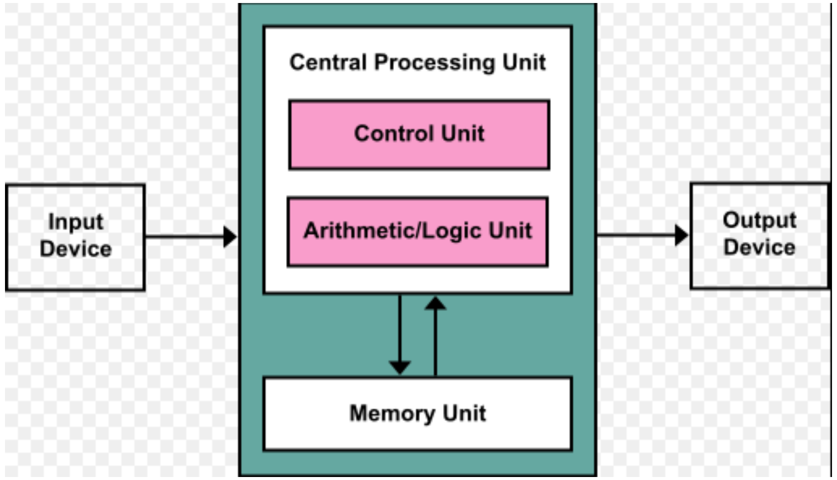
Concerning Von Neumann Architecture, which is the following is incorrect?

A. It has central processing unit

B. It has memory unit

C. It has HDMI cable

D. It has input device



MC

Concerning C++, which is the following is incorrect?

A. Inherent major features of C

B. It is an object-oriented programming language

C. It is is powerful in low level memory manipulation

D. It is usually slower than Python

MC

Concerning C++, which is the following is incorrect?

A. Inherent major features of C

B. It is an object-oriented programming language

C. It is is powerful in low level memory manipulation

D. It is usually slower than Python

MC

Binary number 1001.001 equals to the decimal number

A. 9.125

B. 18.125

C. 9.25

D. 18.25

MC

Binary number $\overset{\overset{8}{\uparrow}}{1}00\overset{\overset{1}{\uparrow}}{1}.00\overset{\overset{0.125}{\nearrow}}{1}$ equals to the decimal number

A. 9.125

B. 18.125

C. 9.25

D. 18.25

MC - more

Hexadecimal to binary

Decimal to hexadecimal

Which of the following is X?

Which of the following is NOT X?

Which of the following is correct/incorrect?

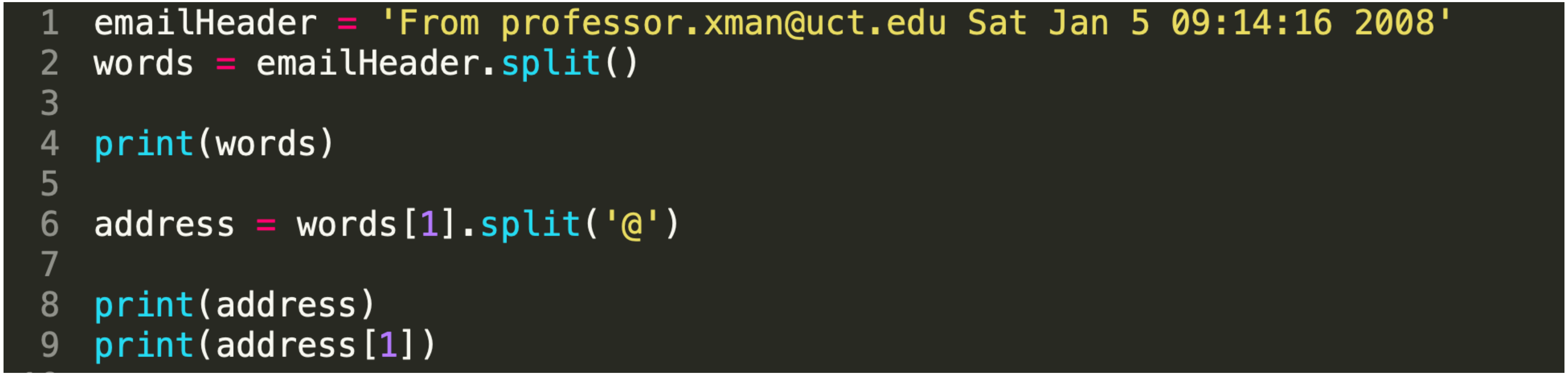
What is the output of the following program?

What is the value of variable x at line 2?

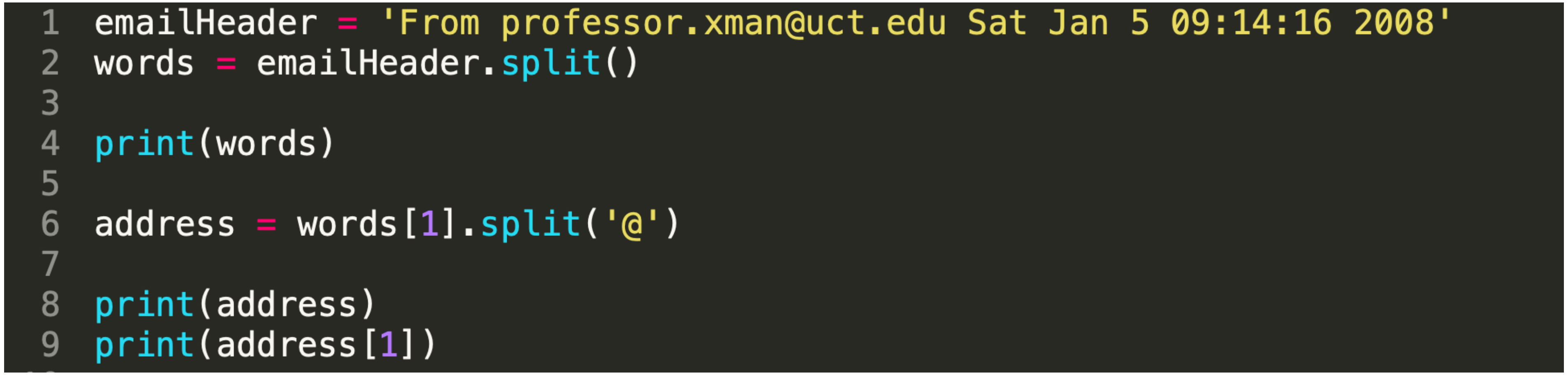
…

Short-answer questions

Read the following program and answer the related questions:



```python
emailHeader = 'From professor.xman@uct.edu Sat Jan 5 09:14:16 2008'
words = emailHeader.split()

print(words)

address = words[1].split('@')

print(address)
print(address[1])
```



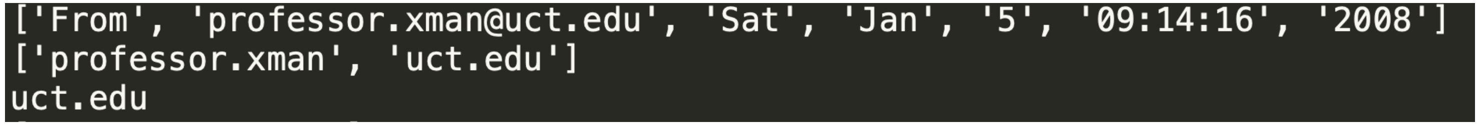
```
1  emailHeader = 'From professor.xman@uct.edu Sat Jan 5 09:14:16 2008'
2  words = emailHeader.split()
3
4  print(words)
5
6  address = words[1].split('@')
7
8  print(address)
9  print(address[1])
```

What can this program do?

What is the data type of variable **words**?

What will be printed at **line 4**?

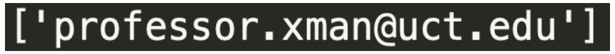
What will be printed at **line 8**?



```
['From', 'professor.xman@uct.edu', 'Sat', 'Jan', '5', '09:14:16', '2008']
['professor.xman', 'uct.edu']
uct.edu
```

What is the value of variable **words** at **line 8**?

What is the difference between **split()** and **split('@')**?



```
['professor.xman@uct.edu']
```

What will be the value of variable **address** if we use **split()** instead of **split('@')** at **line 6**?

What will be the **output** for running **line 9** if we use **split()** instead of **split('@')** at **line 6**?

…

The units of information (data)

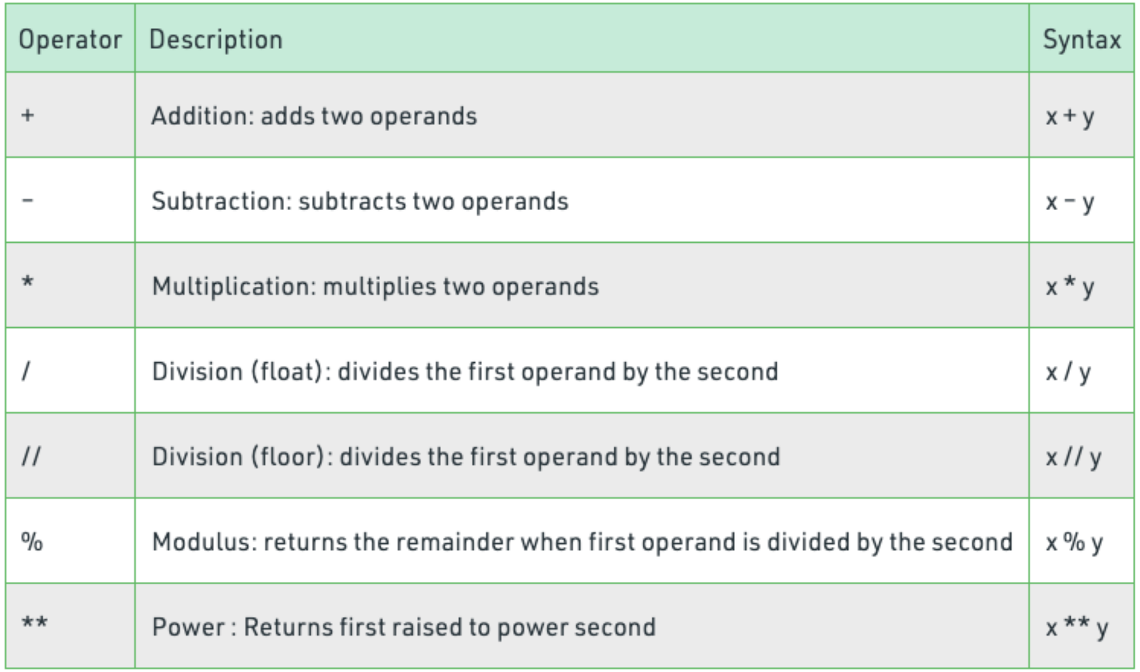
- Bit (比特/位): a binary digit which takes either 0 or 1
- Bit is the smallest information unit in computer programming
- Byte (字节): 1 byte = 8 bits, every English character is represented by 1 byte
- KB (千字节)：1 KB = 2^10 B = 1024 B
- MB (兆字节)：1MB = 2^20 B = 1024 KB
- GB (千兆字节)：1GB = 2^30 B = 1024 MB
- TB (兆兆字节)：1TB = 2^40 B = 1024 GB

All functions introduced

- divmod()
- str()
- input()
- print()
- eval()
- ….

Here we list those mentioned in lec2 as examples.

All operators introduced

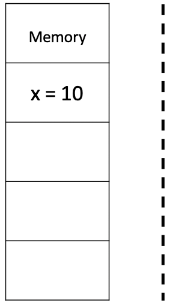


Operator	Description	Syntax
+	Addition: adds two operands	x + y
–	Subtraction: subtracts two operands	x – y
*	Multiplication: multiplies two operands	x * y
/	Division (float): divides the first operand by the second	x / y
//	Division (floor): divides the first operand by the second	x // y
%	Modulus: returns the remainder when first operand is divided by the second	x % y
**	Power : Returns first raised to power second	x ** y

Here we list the arithmetic operators as examples.

Variable

- A variable is a named space in the memory where a programmer can store data and later retrieve the data using the variable name



```
>>> x = 10
>>> x = 20
```

Constants

- Fixed values such as numbers and letters are called constants, since their values won't change
- String constants use single-quotes (') or double-quotes (")

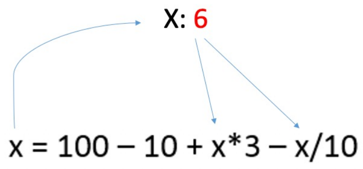
Reserved words

- You cannot use the following words as variables

```
False    None     True     and      as       assert   break
class    continue def      del      elif     else     except
finally  for      from     global   if       import   in
is       lambda   nonlocal not      or       pass     raise
return   try      while    with     yield
```

Assignment statement

- There is a location in the memory for x
- Whenever the value of x is needed, it can be retrieved from the memory
- After the expression is evaluated, the result will be put back into x



x = 100 – 10 + x*3 – x/10

Order evaluation

- When we put operators together, Python needs to know which one to do first
- This is called “operator precedence”
- Which operator “takes precedence” over the others

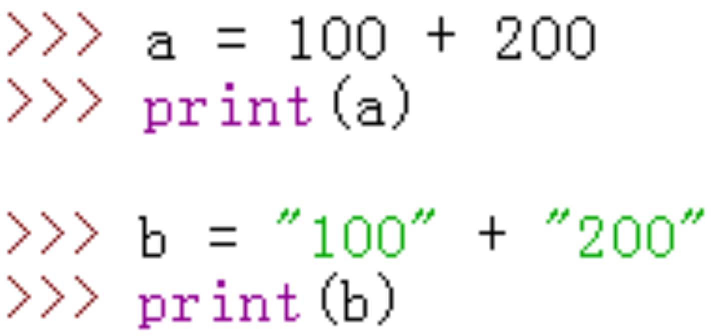
Example: X = 1 + 2*3 – 4 / 5 ** 6

Data Type

- In Python, variables and constants have an associated “type”

- Python knows the difference between a number and a string

- Example:



```
>>> a = 100 + 200
>>> print(a)

>>> b = "100" + "200"
>>> print(b)
```

Comments

- Anything after a “#” is ignored by Python

- Why comment?

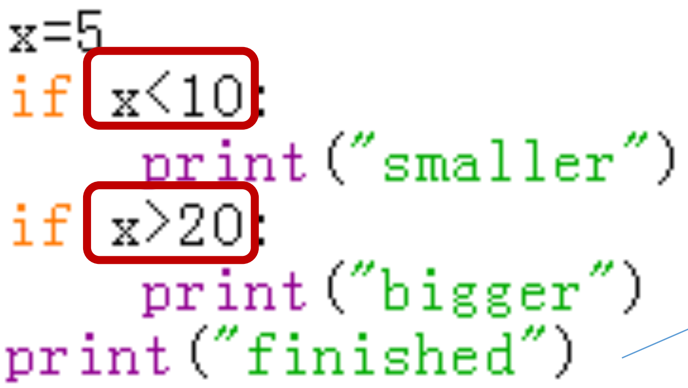
✓Describe what is going to happen in a sequence of code

✓Document who wrote the code and other important information

✓Turn off a line of code – usually temporarily

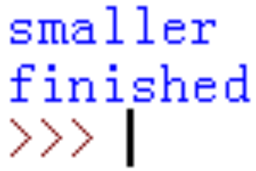
Conditional flow

Program



```
x=5
if x<10:
    print("smaller")
if x>20:
    print("bigger")
print("finished")
```

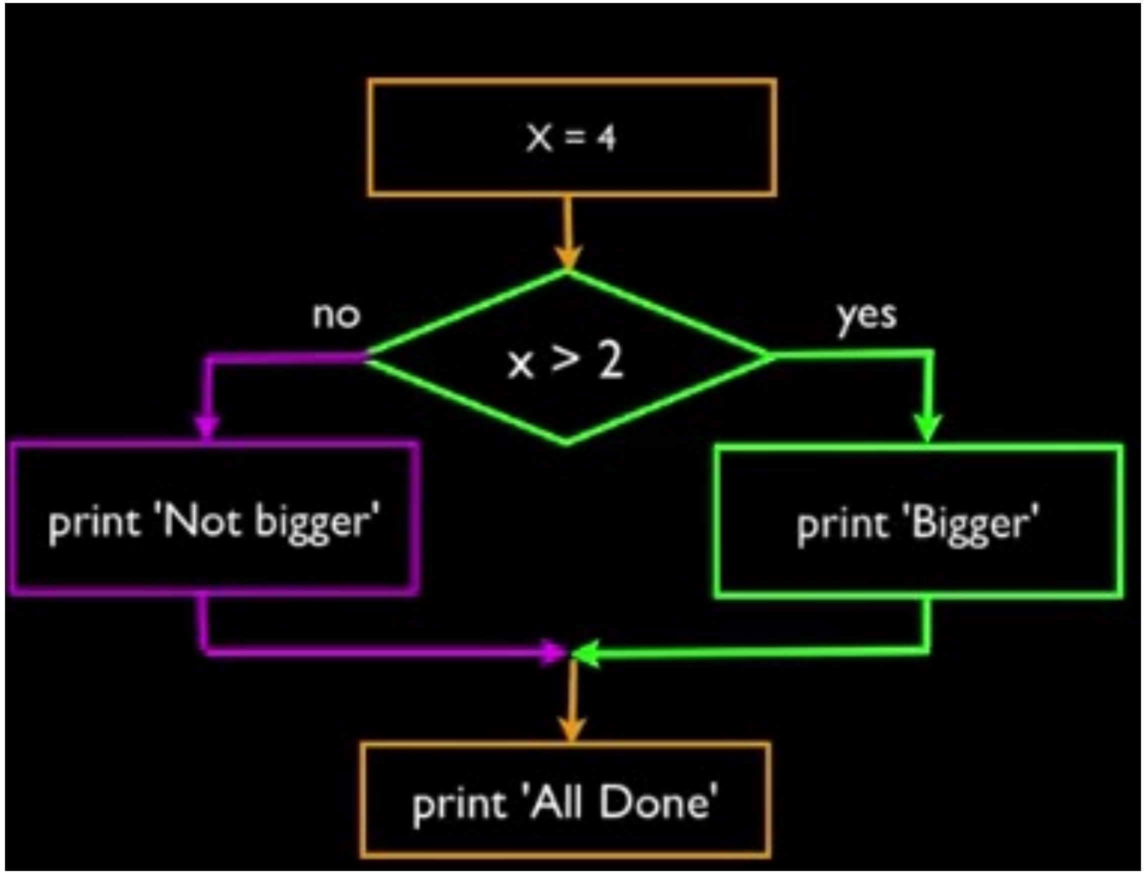
Outputs



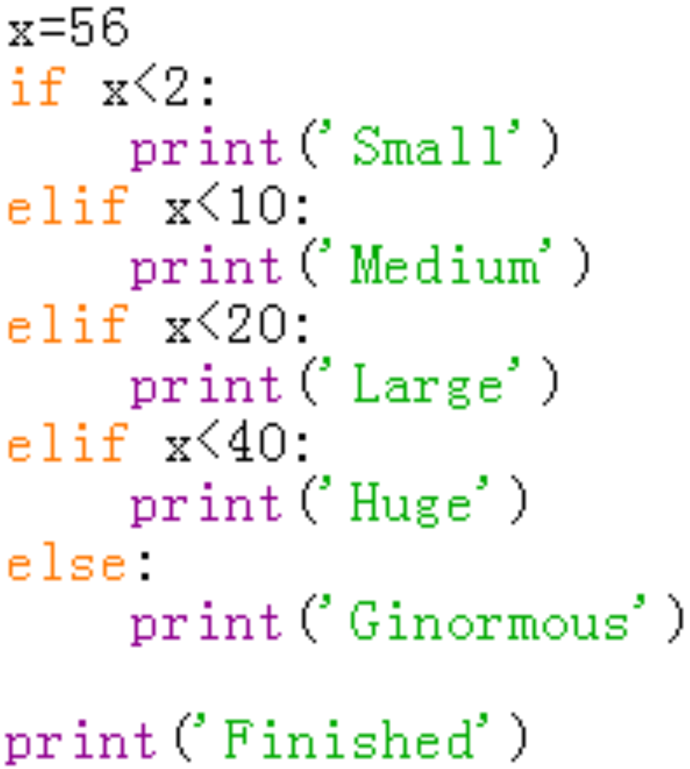
```
smaller
finished
>>> 
```

Two way decision using else

```
x=1

if x>2:
    print(’Bigger’)



print(’Finished’)
```



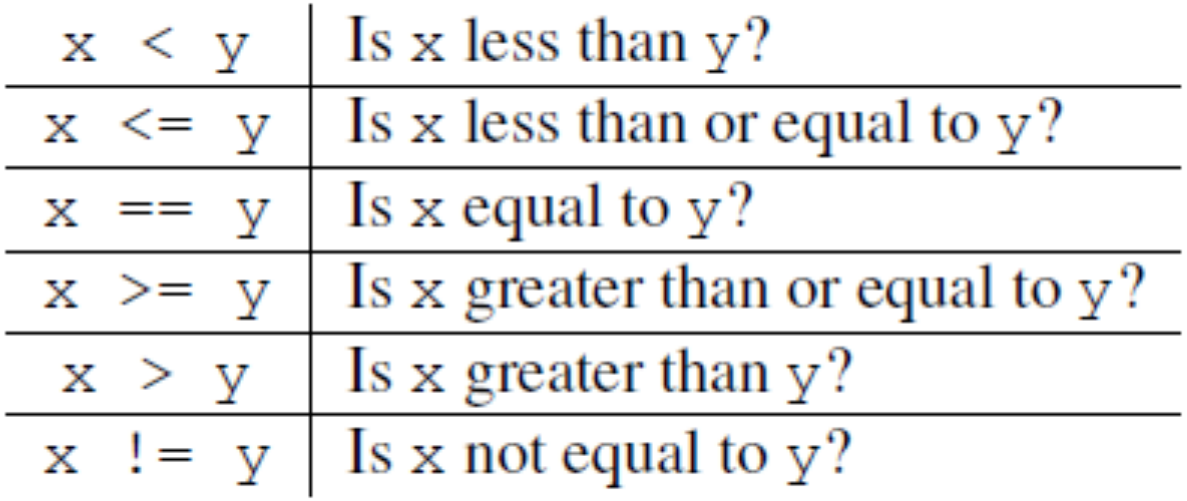
Multi-way decision



```
x=56
if x<2:
    print(’Small’)
elif x<10:
    print(’Medium’)
elif x<20:
    print(’Large’)
elif x<40:
    print(’Huge’)
else:
    print(’Ginormous’)

print(’Finished’)
```

Comparison operators

- Boolean expressions ask a question and produce a Yes/No result, which we use to control program flow
- Boolean expressions use comparison operators to evaluate Yes/No or True/False
- Comparison operators check variables but do not change the values of variables
- Careful!! “=“ is used for assignment



x < y	Is x less than y?
x <= y	Is x less than or equal to y?
x == y	Is x equal to y?
x >= y	Is x greater than or equal to y?
x > y	Is x greater than y?
x != y	Is x not equal to y?

Logical operators

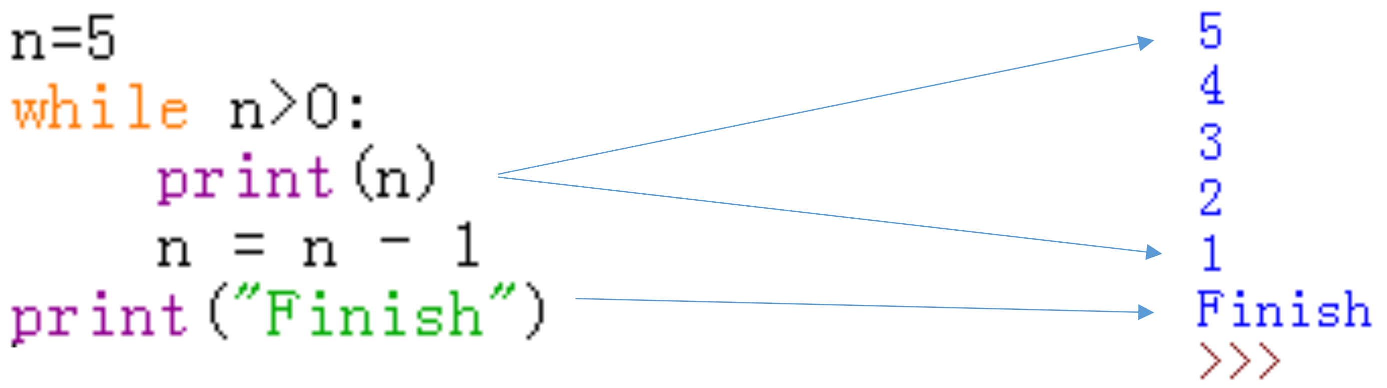
- Logical operators can be used to combine several logical expressions into a single expression

- Python has three logical operators: not, and, or

Repeated flow

Program

Outputs



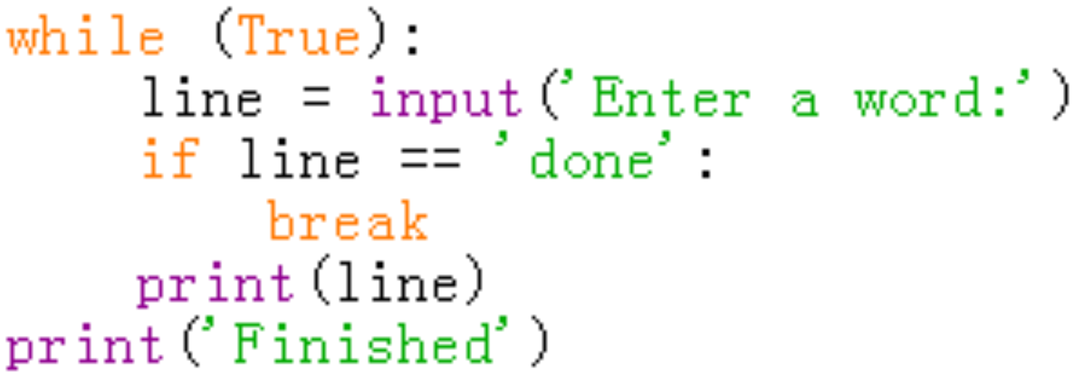
```
n=5
while n>0:
    print(n)
    n = n - 1
print("Finish")
```

```
5
4
3
2
1
Finish
>>>
```

- Loops (repeated steps) have iterative variables that change each time through a loop
- Often these iterative variables go through a sequence of numbers

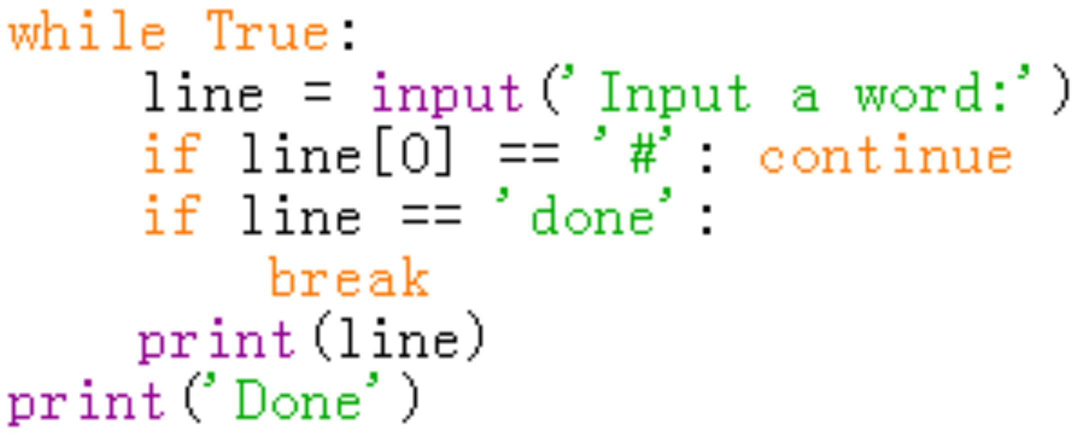
Breaking out of a loop

- The break statement ends the current loop, and jumps to the statement which directly follows the loop



```
while (True):
    line = input(’Enter a word:’)
    if line == ’done’:
        break
    print(line)
print(’Finished’)
```

Finishing an iteration with continue

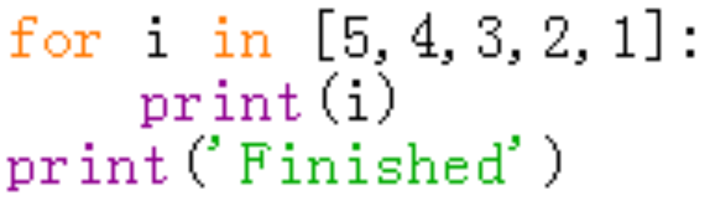


```
while True:
    line = input(’Input a word:’)
    if line[0] == ’#’: continue
    if line == ’done’:
        break
    print(line)
print(’Done’)
```

- The continue statement ends the current iteration, and start the next iteration immediately

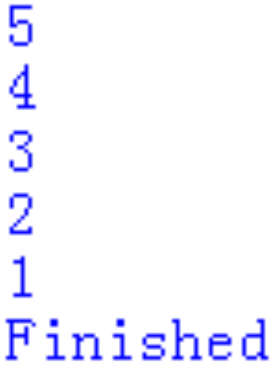
For loop

Example



```
for i in [5,4,3,2,1]:
    print(i)
print(’Finished’)
```

Output

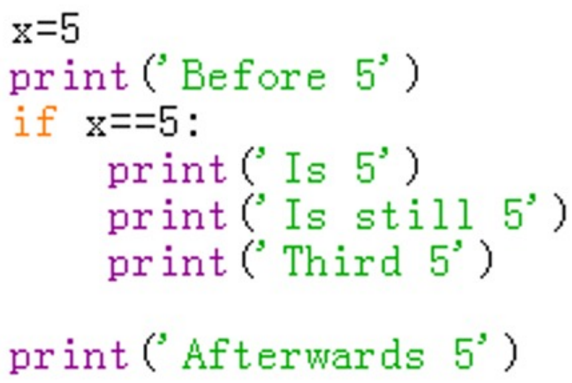


```
5
4
3
2
1
Finished
```

- For loops (definite loops) have explicit iteration variables that change each time through a loop.
- These iteration variables move through a sequence or a set

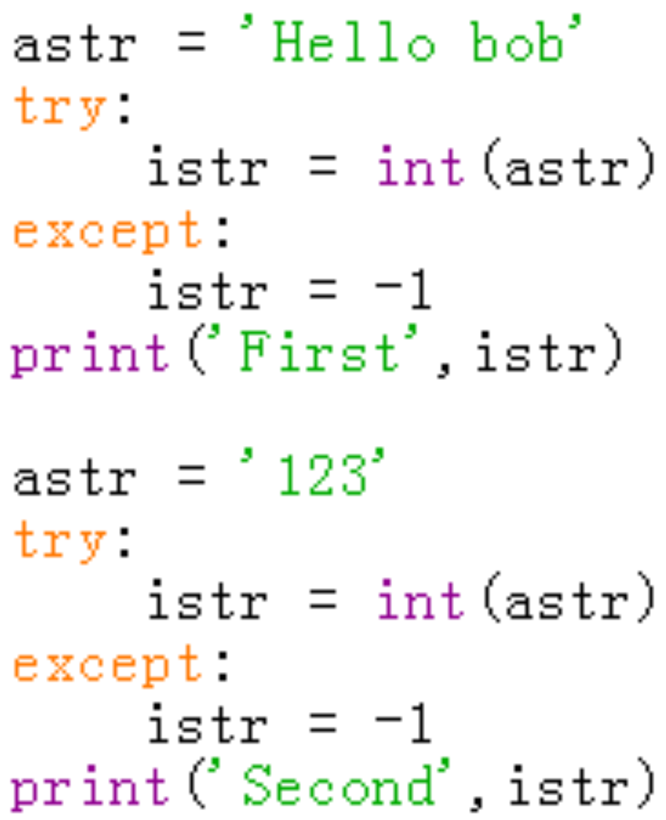
Indentation

- Increase indent: indent after an if or for statement (after :)
- Maintain indent: to indicate the scope of the block (which lines are affected by the if/for)
- Decrease indent: to back to the level of the if statement or for



```
x=5
print(’Before 5’)
if x==5:
    print(’Is 5’)
    print(’Is still 5’)
    print(’Third 5’)

print(’Afterwards 5’)
```

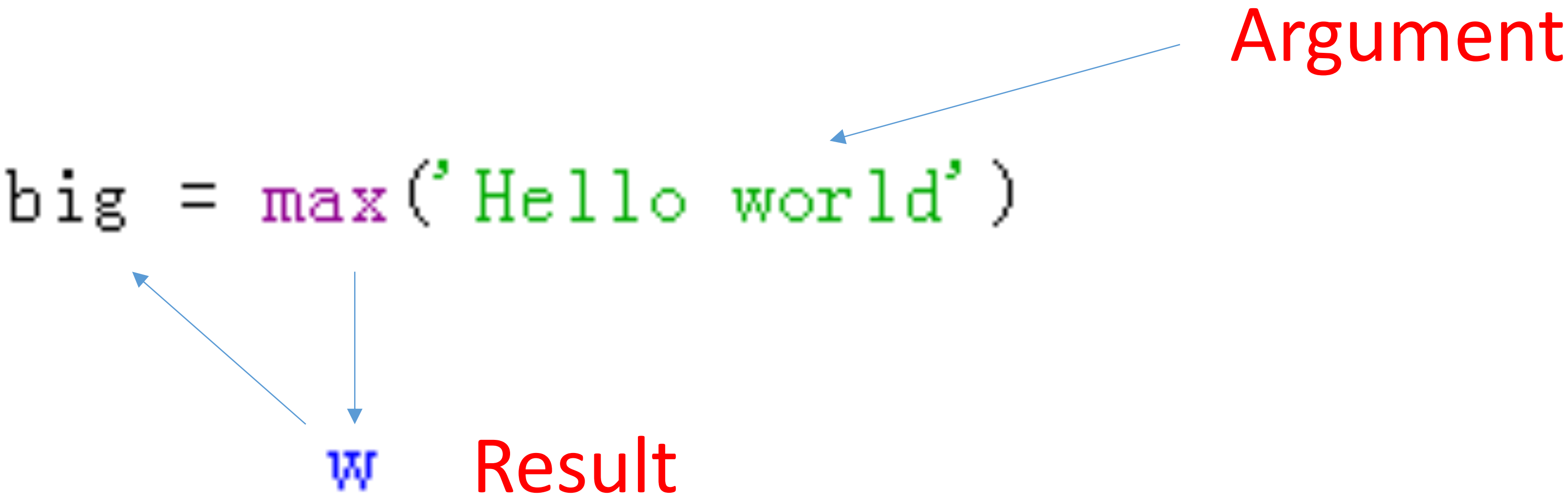
Use try/except to capture errors



```
astr = ’Hello bob’
try:
    istr = int(astr)
except:
    istr = -1
print(’First’,istr)

astr = ’123’
try:
    istr = int(astr)
except:
    istr = -1
print(’Second’,istr)
```

- When the first conversion fails, it just stops into the except block, and the program continues

- When the second conversion succeeds, it just skips the except block

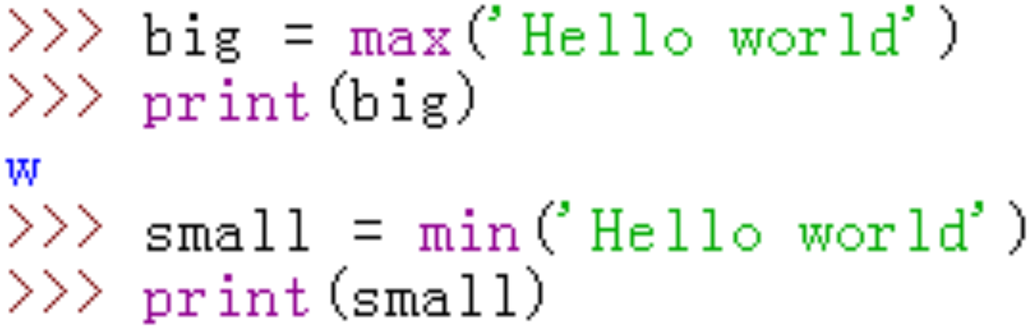


Argument

```
big = max(’Hello world’)
```

```
w
```

Result



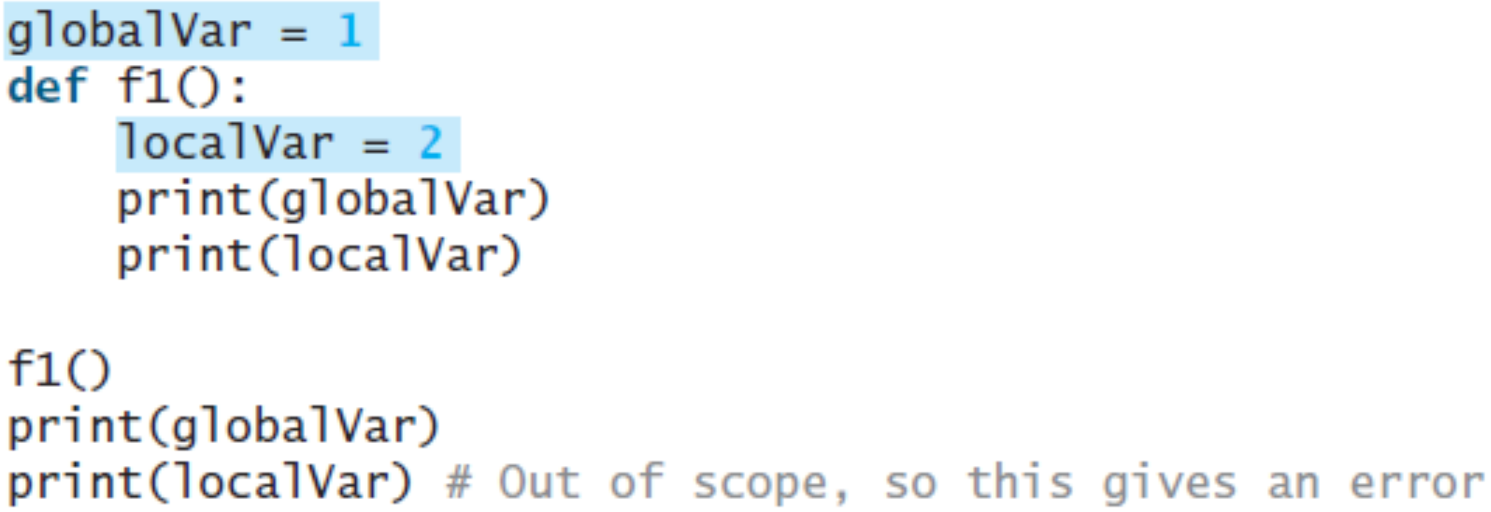
```
>>> big = max(’Hello world’)
>>> print(big)
w
>>> small = min(’Hello world’)
>>> print(small)
```

Scope of variables

- The scope of a variable is the part of program where this variable can be accessed

- A variable created inside a function is referred to as a local variable

- Global variables are created outside all functions and are accessible to all functions in their scope



```
globalVar = 1
def f1():
    localVar = 2
    print(globalVar)
    print(localVar)

f1()
print(globalVar)
print(localVar) # Out of scope, so this gives an error
```

Rules for defining variables in Python

- Must start with a letter or underscore _

  courseName **_courseName**

- Can only contain letters, numbers and underscore

  course~Name

- Case sensitive

  courseName **coursename**

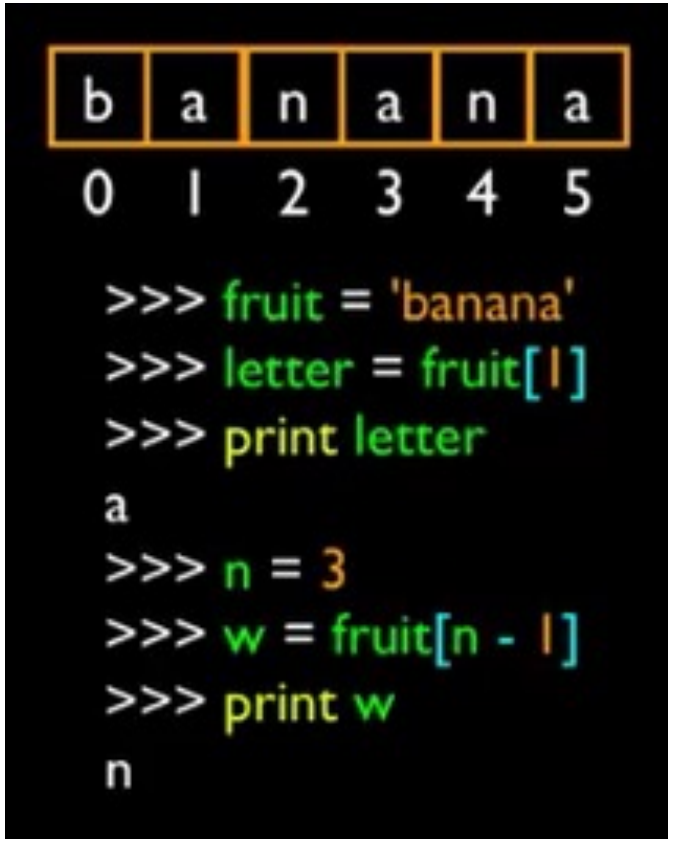
- Good: apple, car, myNumber123, _light
- Bad: 456aaa, #ab, var.12
- Different: apple, Apple, APPLE

Looking inside strings

- We can get any character in a string using an index specified in square brackets

- The index value must be an integer which starts from zero

- The index value can be an expression



b	a	n	a	n	a
0	1	2	3	4	5

```
>>> fruit = 'banana'
>>> letter = fruit[1]
>>> print letter
a
>>> n = 3
>>> w = fruit[n - 1]
>>> print w
n
```

String operations

- Some operators apply to strings

✓“+”: concatenation

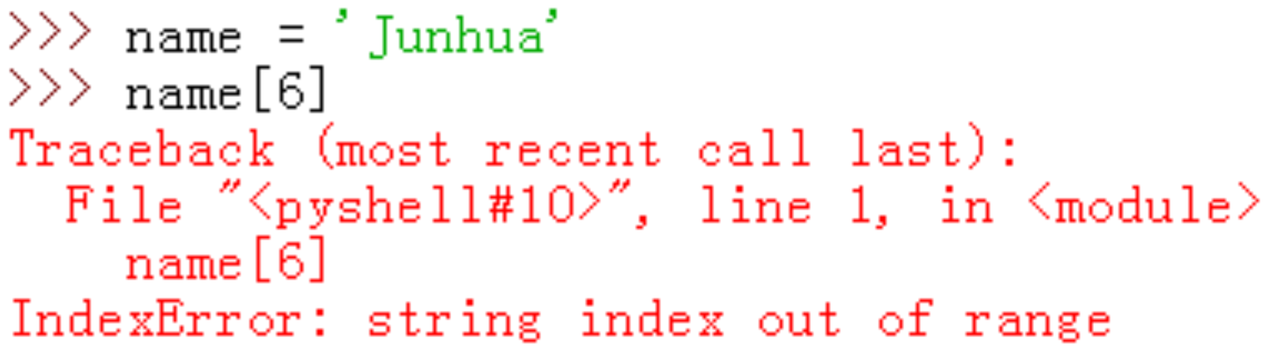
✓“*”: multiple concatenation

- Python knows whether it is dealing with a number or a string

Index out of range

- You will get an Python error if you attempt to index beyond the end of a string

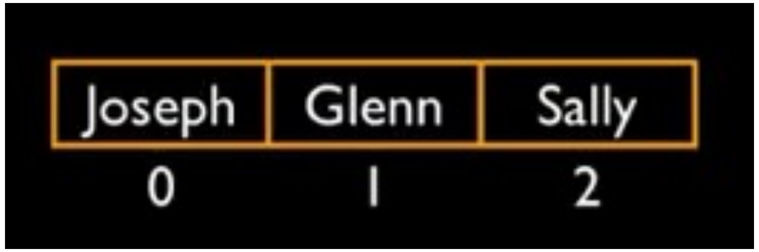
- Be careful when specifying an index value



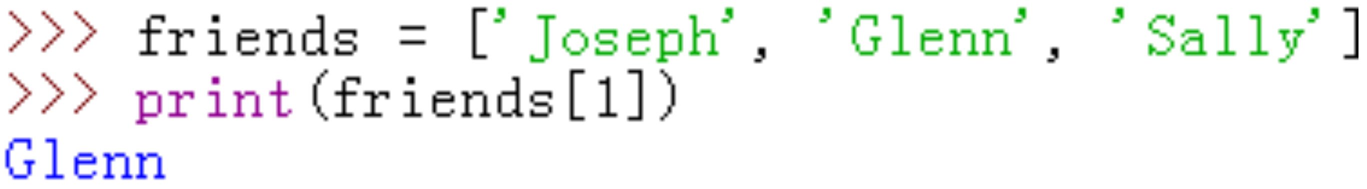
```
>>> name = 'Junhua'
>>> name[6]
Traceback (most recent call last):
  File "<pyshell#10>", line 1, in <module>
    name[6]
IndexError: string index out of range
```

Looking inside lists

- Just like strings, we can access any single element in a list using an index specified in square bracket



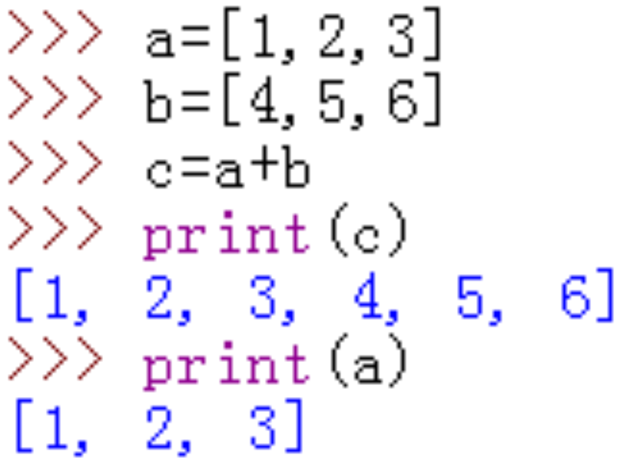
Joseph	Glenn	Sally
0	1	2



```
>>> friends = ['Joseph', 'Glenn', 'Sally']
>>> print(friends[1])
Glenn
```

Concatenating lists using +

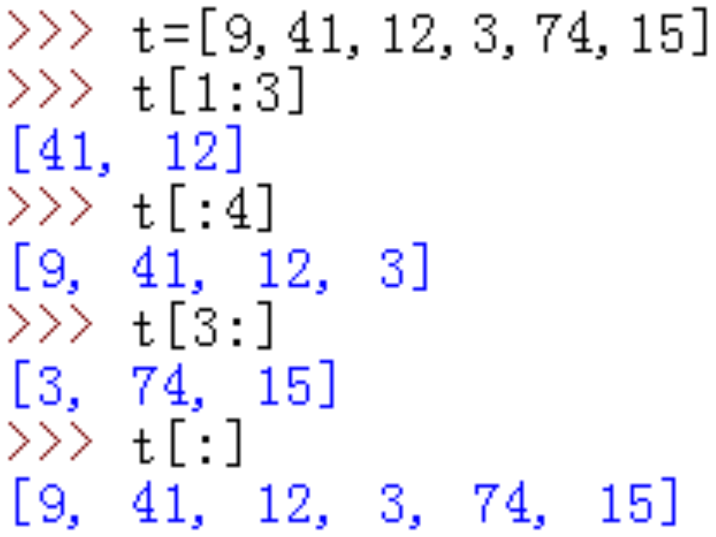
- Similar to strings, we can add two existing lists together to create a new list



```
>>> a=[1,2,3]
>>> b=[4,5,6]
>>> c=a+b
>>> print(c)
[1, 2, 3, 4, 5, 6]
>>> print(a)
[1, 2, 3]
```

Lists can be sliced using :

- Remember: similar to strings, the second number is “up to but no including”



```
>>> t=[9,41,12,3,74,15]
>>> t[1:3]
[41, 12]
>>> t[:4]
[9, 41, 12, 3]
>>> t[3:]
[3, 74, 15]
>>> t[:]
[9, 41, 12, 3, 74, 15]
```

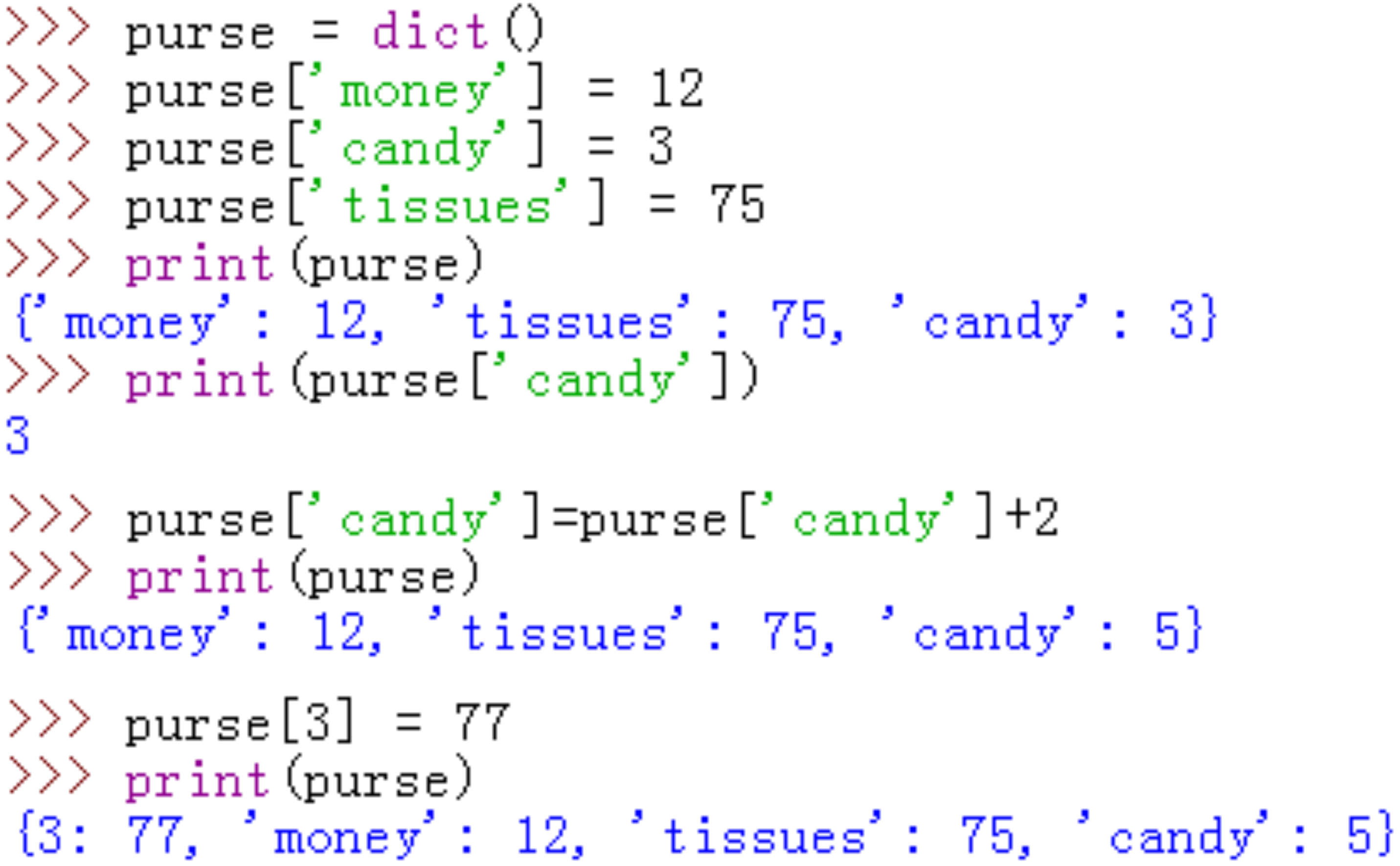
https://stackoverflow.com/questions/509211/understanding-slicing

Dictionary

- Lists index their entries based on the position in the list

- Dictionaries are like bags – no order

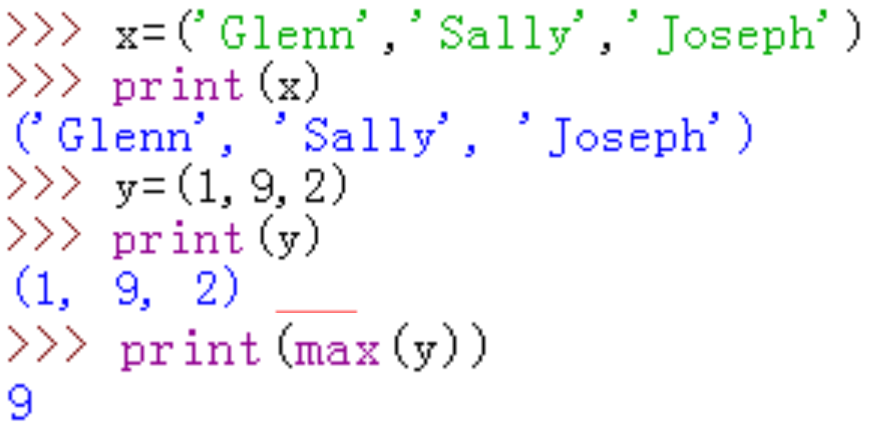
- We index the elements we put in the dictionary with a “lookup tag”



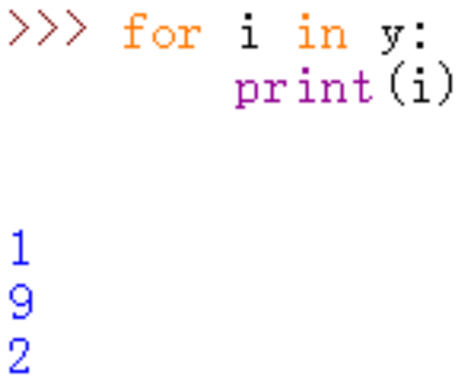
```
>>> purse = dict()
>>> purse['money'] = 12
>>> purse['candy'] = 3
>>> purse['tissues'] = 75
>>> print(purse)
{'money': 12, 'tissues': 75, 'candy': 3}
>>> print(purse['candy'])
3

>>> purse['candy']=purse['candy']+2
>>> print(purse)
{'money': 12, 'tissues': 75, 'candy': 5}

>>> purse[3] = 77
>>> print(purse)
{3: 77, 'money': 12, 'tissues': 75, 'candy': 5}
```

Tuples

- Tuples are another type of sequence that function more like a list – they have elements which are indexed starting from 0



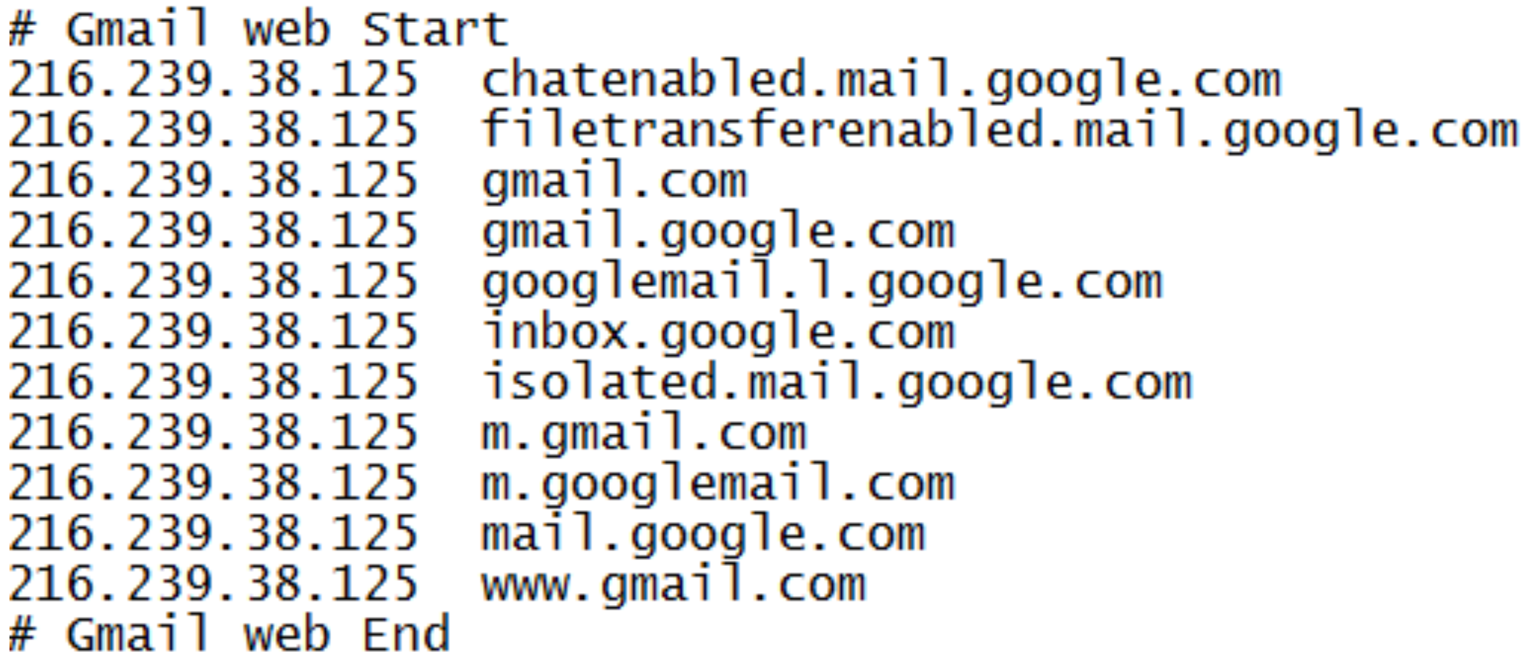
```
>>> x=('Glenn','Sally','Joseph')
>>> print(x)
('Glenn', 'Sally', 'Joseph')
>>> y=(1,9,2)
>>> print(y)
(1, 9, 2)
>>> print(max(y))
9
```



```
>>> for i in y:
        print(i)


1
9
2
```

File processing

- A text file can be thought of as a sequence of lines
- A text file has newline at the end of each line

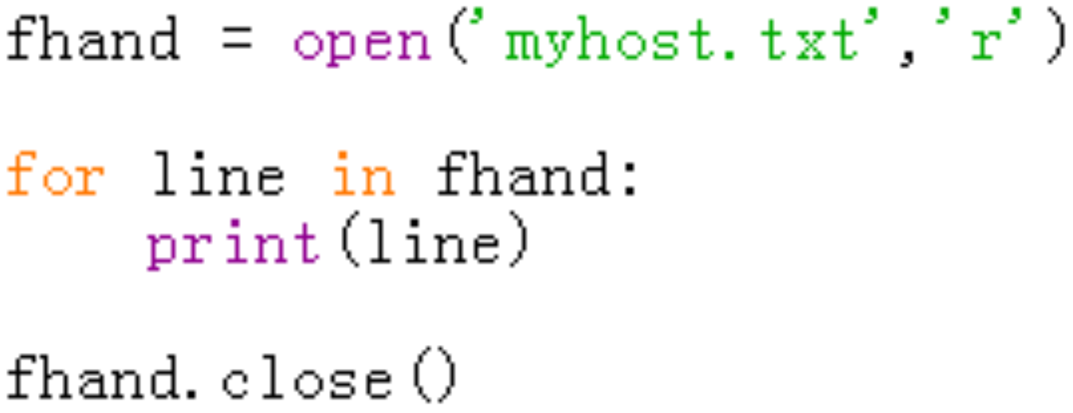


```
# Gmail web Start
216.239.38.125  chatenabled.mail.google.com
216.239.38.125  filetransferenabled.mail.google.com
216.239.38.125  gmail.com
216.239.38.125  gmail.google.com
216.239.38.125  googlemail.l.google.com
216.239.38.125  inbox.google.com
216.239.38.125  isolated.mail.google.com
216.239.38.125  m.gmail.com
216.239.38.125  m.googlemail.com
216.239.38.125  mail.google.com
216.239.38.125  www.gmail.com
# Gmail web End
```

File handle as a sequence

- A file handle open for read can be treated as a sequence of strings where each line in the file is a string in the sequence

- We can use the for statement to loop through a sequence



```
fhand = open(’myhost.txt’,’r’)

for line in fhand:
    print(line)

fhand.close()
```