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

Introduction to Computer Science: Programming Methodology

Lecture 2 Python Basics

Prof. Pinjia He

School of Data Science

Parseltongue is the language of serpents and those who can converse with them.

蛇语者

An individual who can speak Parseltongue is known as a Parselmouth.

It is very uncommon skill, and may be hereditary. Nearly all known Parselmouths are descended from Salazar Slytherin.

http://harrypotter.wikia.com/wiki/Parseltongue

Python is the language of Python interpreter and those who can converse with them.

An individual who can speak Python is known as a Pythonista.

It is very uncommon skill, and may be hereditary. Nearly all known Pythonistas use software initially developed by Guido van Rossum

Interpreter

- Interpreter (解释器) is a computer program that directly executes, i.e. performs, instructions written in a programming or scripting language, without previously compiling them into a machine language program

Interpreter v.s. compiler

直接执行用编程语言

- Interpreter (解释器) is a computer program that directly executes, i.e. performs, instructions written in a programming or scripting language, without previously compiling them into a machine language program

编写指令的程序

- A compiler (编译器) is a computer program (or a set of programs) that transforms source code written in a programming language (the source language) into another computer language (the target language), with the latter often having a binary form known as object code

把源代码转换成低级语言.

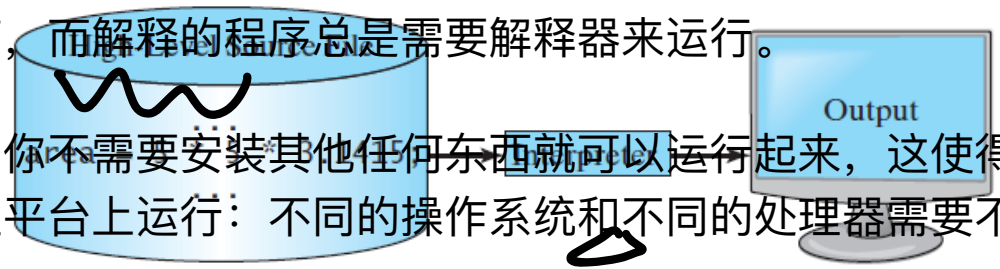
compiler

编译器是把源程序的每一条语句都编译成机器语言,并保存成二进制文件,这样运行时计算机可以直接以机器语言来运行此程序,速度很快;

Interpreter v.s. compiler

解释器则是只在执行程序时,才一条一条的解释成机器语言给计算机来执行,所以运行速度是不如编译后的程序运行的快的。 interpreter

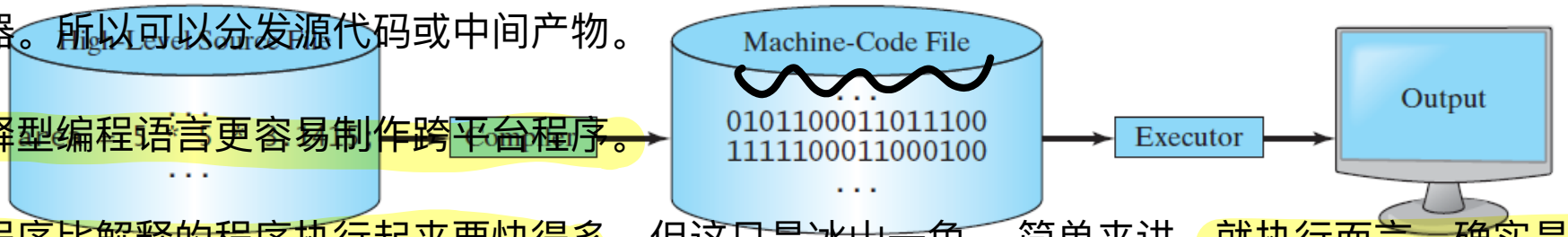
编译器生成一个独立的程序，而解释的程序总是需要解释器来运行。



(a)

如果你有一个编译的程序，你不需要安装其他任何东西就可以运行起来，这使得分发十分简单。 另一方面，可执行文件在一个特定平台上运行：不同的操作系统和不同的处理器需要不同的编译版本。

如果要解释程序，可以将不同平台上的相同副本分发给用户。 然而，他们需要一个在其特定平台上运行的解释器。所以可以分发源代码或中间产物。



(b)

使用解释型编程语言更容易制作跨平台程序。

编译的程序比解释的程序执行起来要快得多，但这只是冰山一角。 简单来讲，就执行而言，确实是编译后执行的编译型程序执行的快些，但是编译型程序的编译加执行的时间比解释性语言解释执行的时间多。

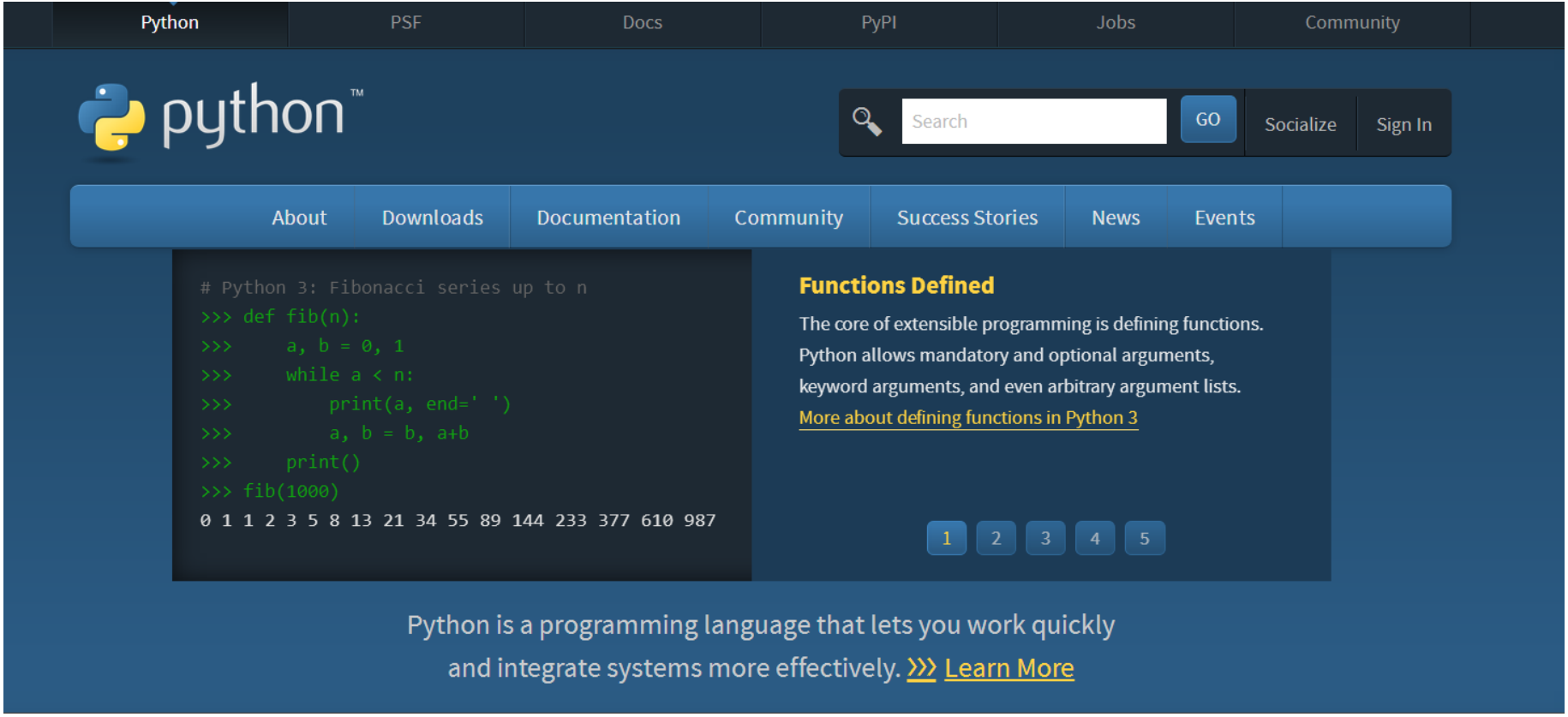
Is Python interpreted or compiled?

https://nedbatchelder.com/blog/201803/is_python_interpreted_or_compiled.html

Early learner: syntax error

- We need to learn the Python language so we can communicate our instructions to Python. In the beginning we will make lots of mistakes and speak gibberish like small children
- When you make a mistake, the computer does not think you are “cute”. It says “syntax error” – given that it “knows” the language and you are just learning it. It seems like Python is cruel and unfeeling
- You must remember that you are intelligent and can learn, while the computer is simple and very fast – but cannot learn
- It is easier for you to learn Python than for the computer to learn human language

Installing Python

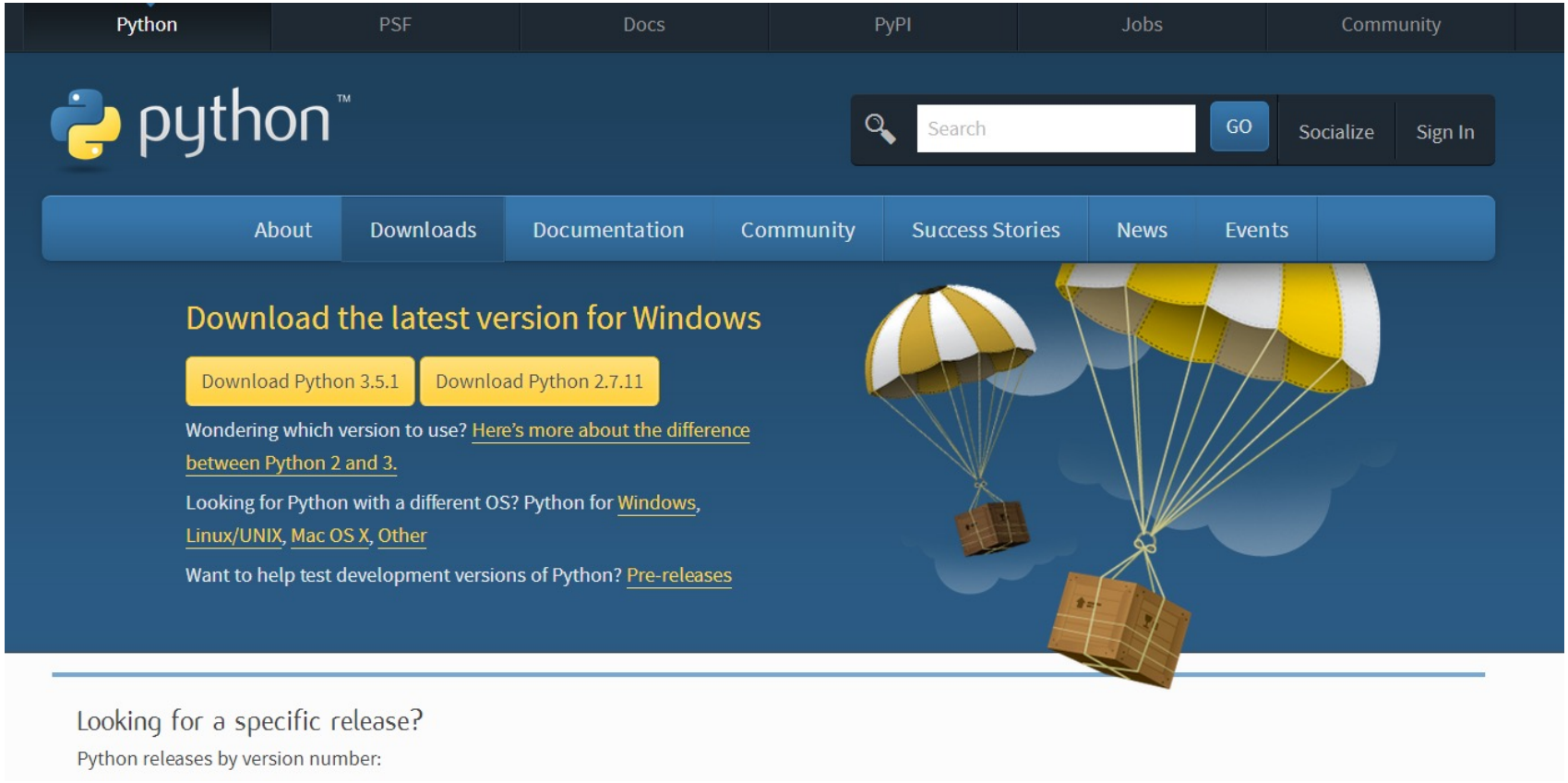


https://www.python.org

Installing Python

Visual Studio Code

PyCharm

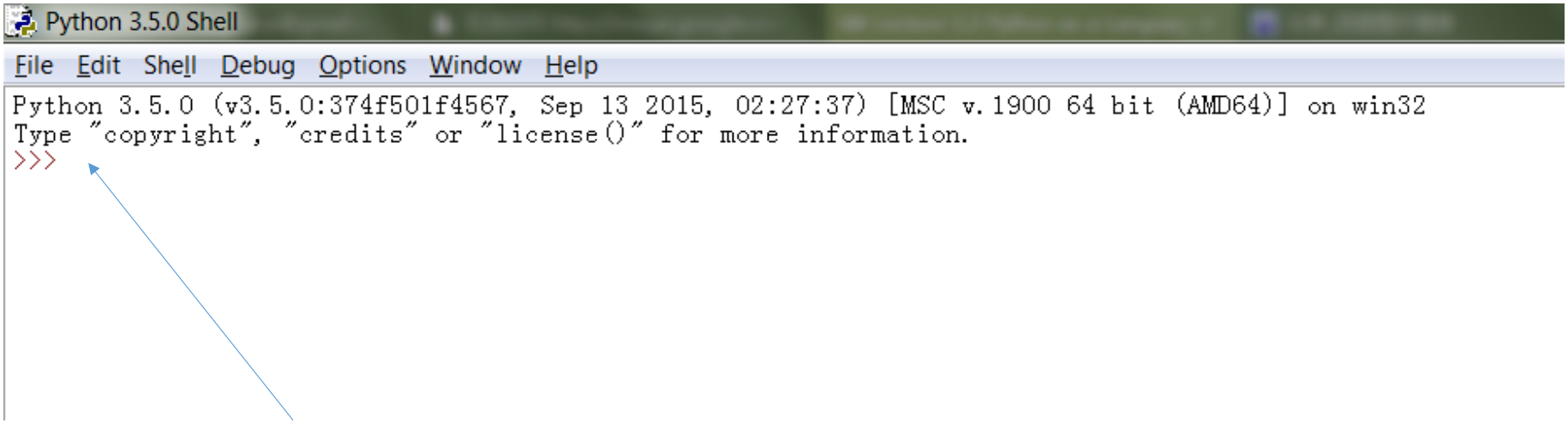


Pycharm和Python关系简单来说：Pycharm是一个代码编辑器，是比较流行的代码编辑器之一，用于编写python代码。也就是说Python是我们进行项目开发而使用的一门计算机语言，为了更好的调试代码和运行，使用界面程序Pycharm进行操作。

Python 3 v.s. Python 2 ?

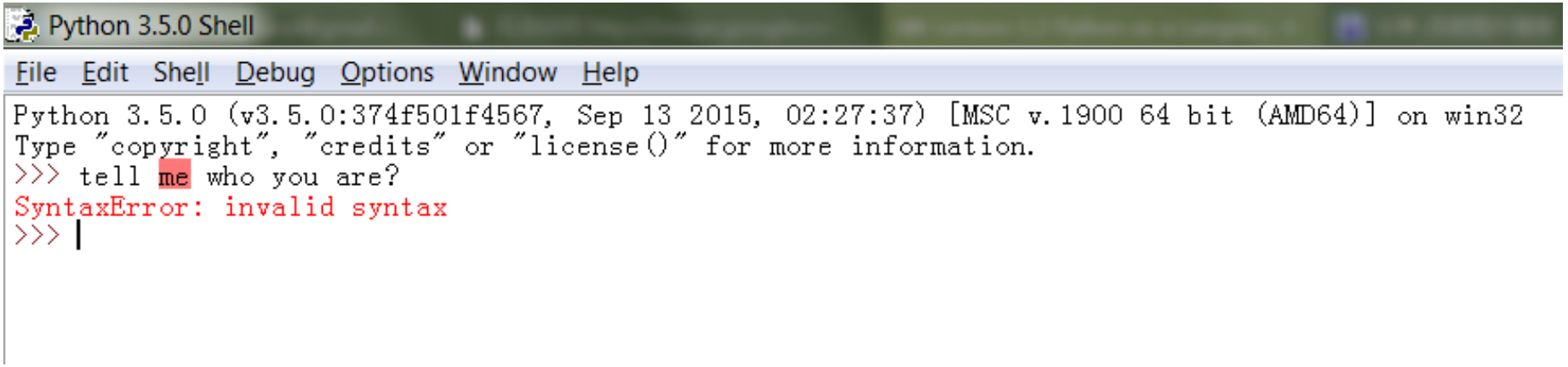
https://learntocodewith.me/learn/python-2-vs-python-3/

Python Shell

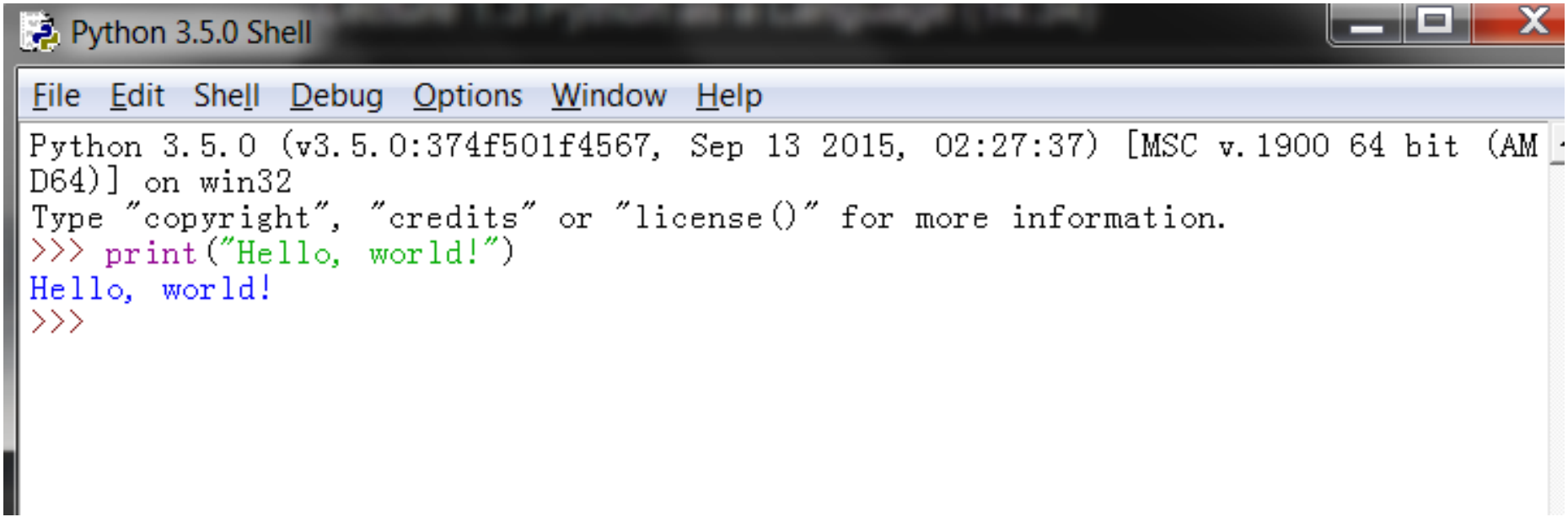


What is next?

Syntax Error



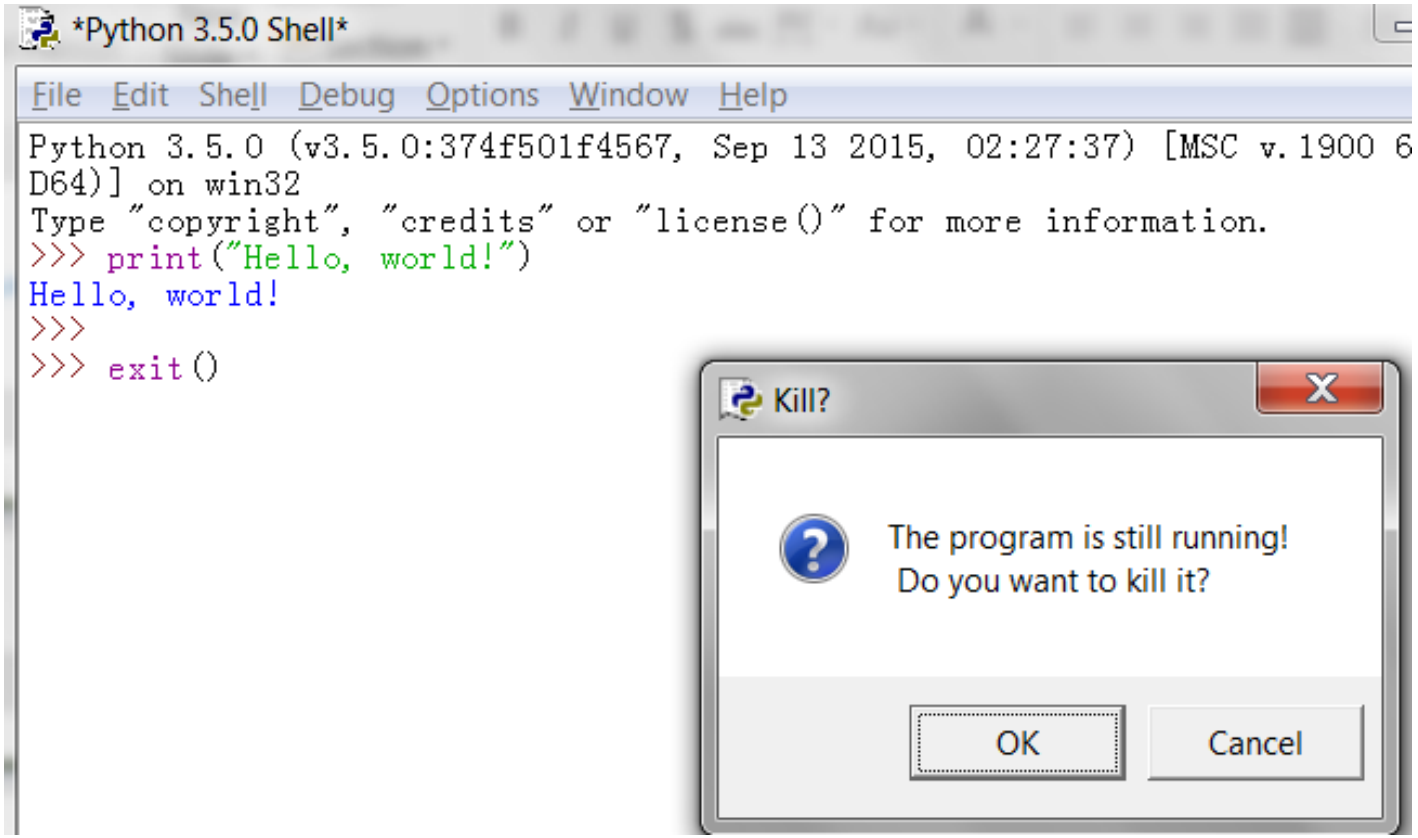
Hello, world!



- You must say something that Python interpreter can understand!!

- Print() is a function in Python

Exit()

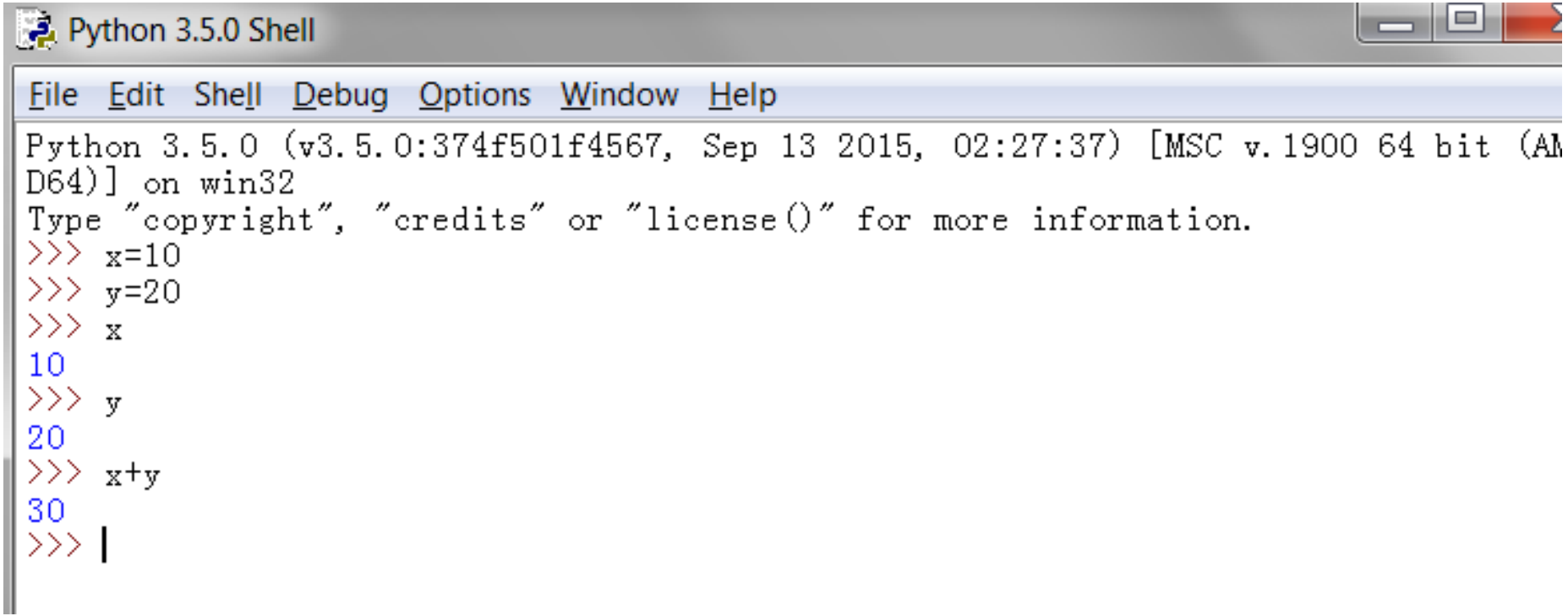

Python 3.5.0 Shell
File Edit Shell Debug Options Window Help
Python 3.5.0 (v3.5.0:374f501f4567, Sep 13 2015, 02:27:37) [MSC v.1900 6
D64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> print("Hello, world!")
Hello, world!
>>>
>>> exit()
Kill?
The program is still running!
Do you want to kill it?
OK
Cancel

What should we say to Python ?

Elements of Python Language

- Vocabulary/words – Variables and Reserved words
- Sentence structure – valid syntax patterns
- Story structure – constructing a meaningful program for some purposes

Variables 变量



(高级语言中已定义过的字).

Reserved words

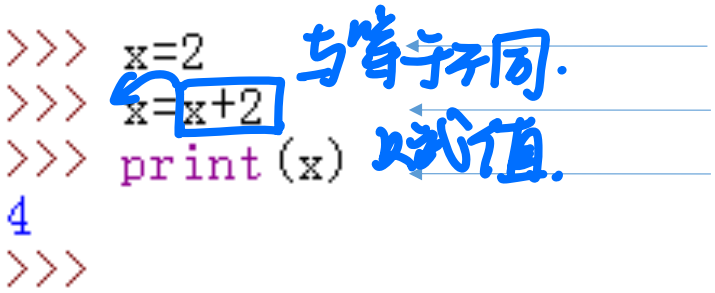
- You cannot use the following words as variables

```
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
```

变量名只能包含字母（大小写均可）、数字和下划线（`_`）。
变量名必须以字母或下划线开头，不能以数字开头。
变量名是区分大小写的，例如`myVar`和`myvar`是两个不同的变量名。

case-sensitive.

Sentences or lines



Assignment statement

Assignment with expressions

Print statement (output statement)

Programming scripts

交互式 python

- Interactive Python is good for experiments and programs of 3-4 lines long

- Most programs are much longer, so we have to type them in a file and execute them all together

- In this sense, we are giving Python a script

作为一个后缀

- As convention, ".py" is added as the suffix on the end of these files

Interactive v.s. script

- Interactive

交互式

✓ You type directly to Python one line at a time and it responds

- Script

✓ You enter a sequence of statements (lines) into a file using a text editor and tell Python to execute the file

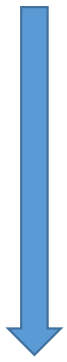
文本编辑器、

Program steps or program flow

- Like a recipe, a program is a sequence of steps to be done in pre-determined order
- Some steps are conditional, i.e. they may be skipped
- Sometimes, we will repeat some steps
- Sometimes, we store a set of steps to be used over and over again in future as needed

Sequential flow 顺序流.

Execute sequentially



```
>>> x=2
>>> print(x)
2
>>> x=x*10
>>> print(x)
20
>>>
```

Outputs

- When a program is running, it flows from one step to the next
- We as programmers, set up “paths” for the program to follow

Conditional flow 条件.

Program

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

Outputs

```
smaller
finished
>>>
```

变式：x=15

⋮

output: finished.

while 循环.

Repeated flow

Program

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

Outputs

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

Q: 为什么调换顺序?

直到 n=0 时停止.

Q2: ?

pn;

交互变量.

A:

- Loops (repeated steps) have iterative variables that change each time through a loop

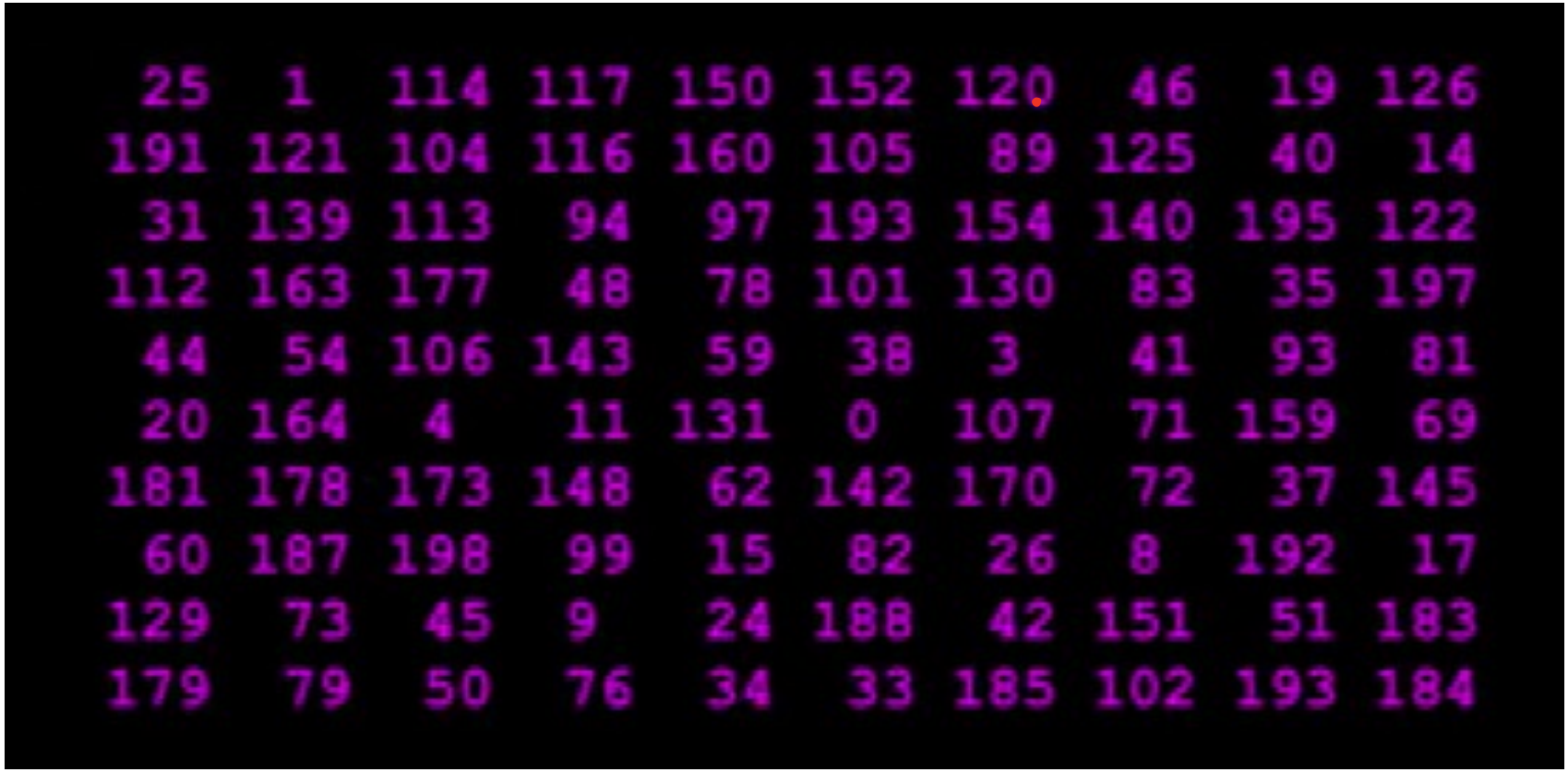
※ while 循环可以帮助得到

- Often these iterative variables go through a sequence of numbers

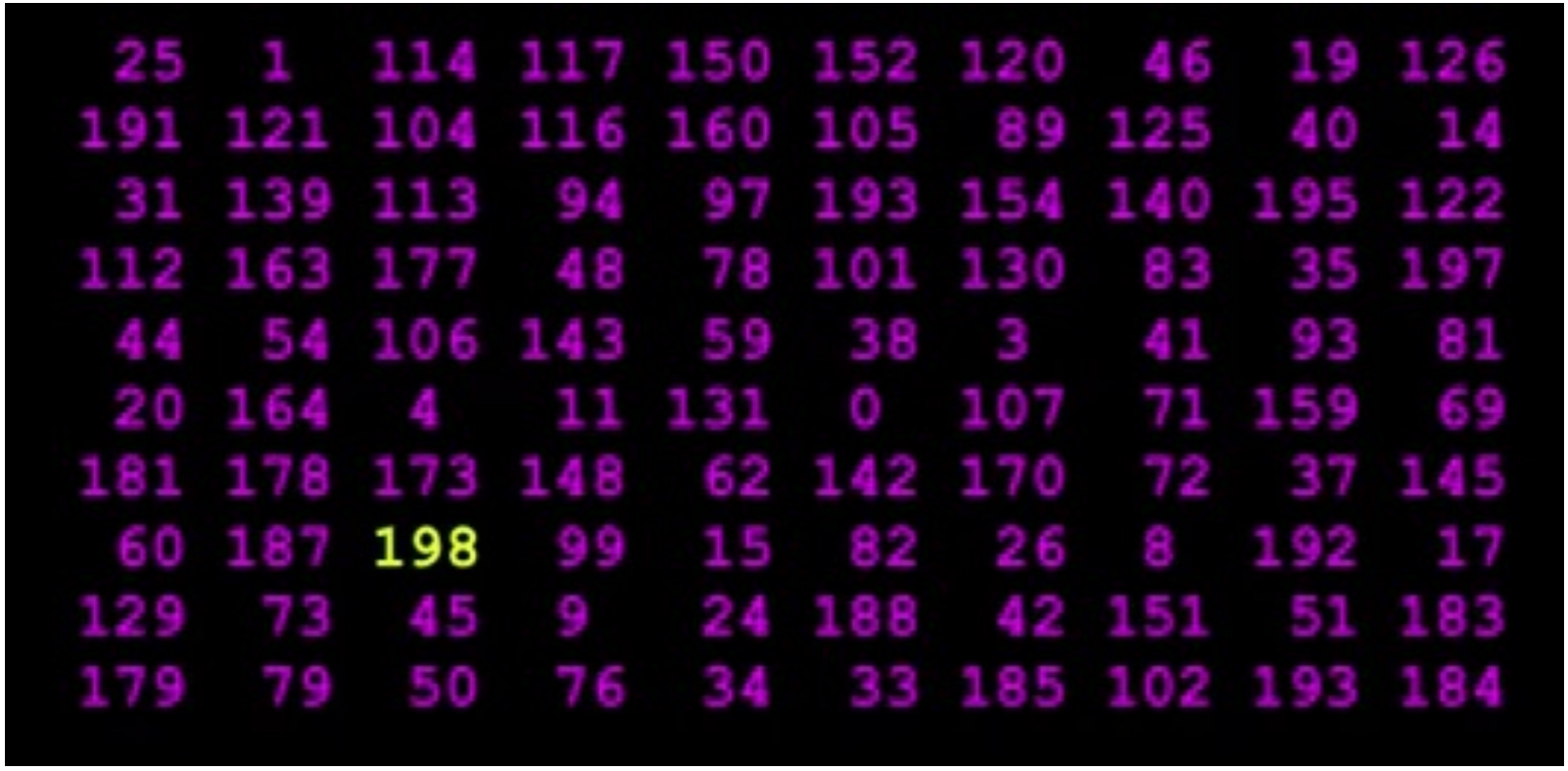
output 将变成 4 3 2 1.

想要的东西.

What the largest number is?



What the largest number is?



Constants 定值.

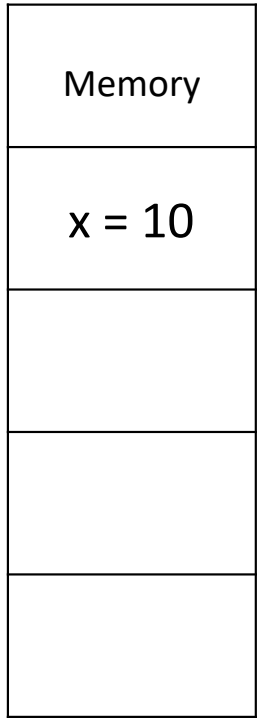
- Fixed values such as numbers and letters are called constants, since their values won't change

- String constants use single-quotes (') or double-quotes (")

字符串常量.

Variable 变量.

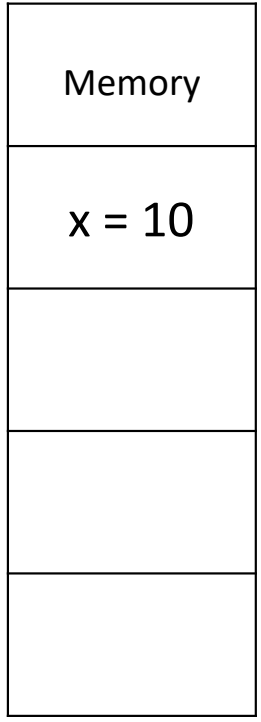
- A variable is a named space in the memory where a programmer can store data and later retrieve the data using the variable name




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

Variable

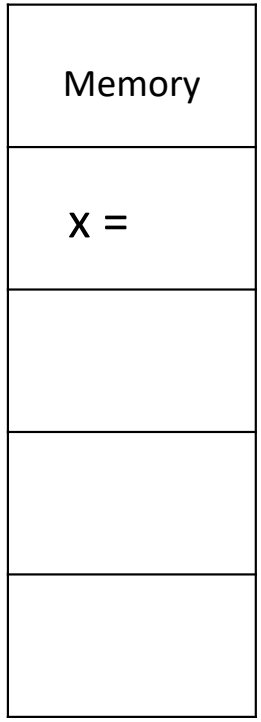
- A variable is a named space in the memory where a programmer can store data and later retrieve the data using the variable name




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

Variable

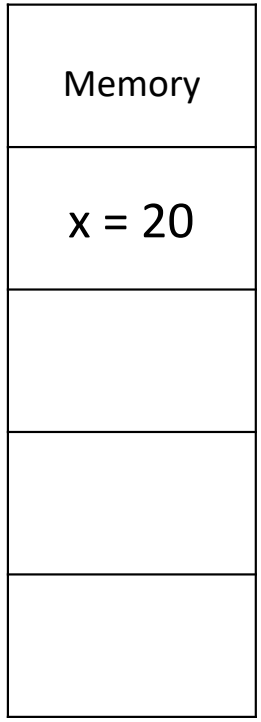
- A variable is a named space in the memory where a programmer can store data and later retrieve the data using the variable name




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

Variable

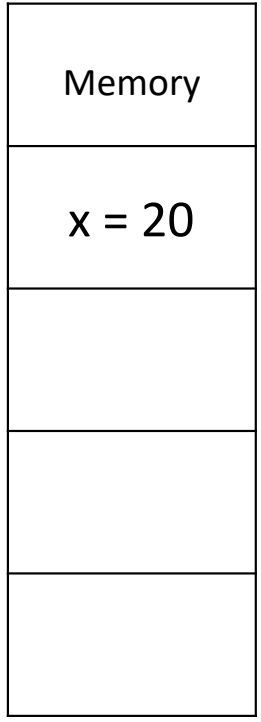
- A variable is a named space in the memory where a programmer can store data and later retrieve the data using the variable name




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

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
>>> print (x)
20
```

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
- Variable names are determined by programmers
- The value of a variable can be changed later in a program

https://www.youtube.com/watch?v=p3q5zWCw8J4

Rules for defining variables in Python

- Must start with a letter or underscore _

  courseName _courseName

- Can only contain letters, numbers and underscore

  course~Name X

- Case sensitive

  courseName coursename

- Good: apple, car, myNumber123, _light

- Bad: 456aaa, #ab, var.12

- Different: apple, Apple, APPLE

start with letters / _ v
(X numbers)

Personal tips

- Use meaningful words as variable names
- Start with a lower letter
- Capitalize the first letter of each word
- Example: myBankAccountID, numOfCards, salaryAtYear1995…

What is this code doing?

```
x1q3z9ocd = 35.0
x1q3z9afd = 12.50
x1q3p9afd = x1q3z9ocd * x1q3z9afd
print x1q3p9afd
```

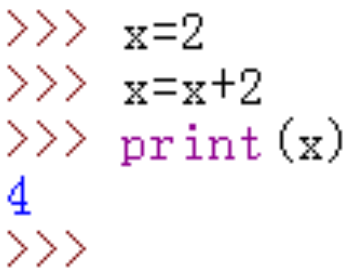
Reserved words 保留字

- 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
```

self, object, ...

Sentences or lines



Assignment statement

Assignment with expressions

Print statement (output statement)

Variable Operator Constant Reserved words

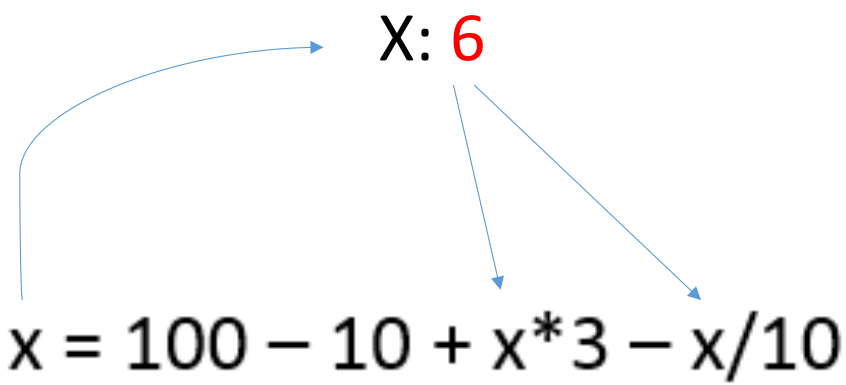
Assignment statement

- We assign a value to a variable using the assignment operator (=)
- An assignment statement consists of an expression on the right hand side, and a variable to store the result

Example: x = 100 – 10 + x*3 – x/10

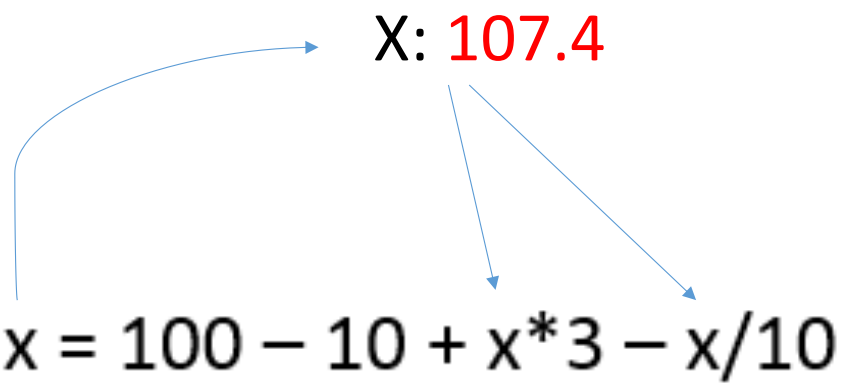
Assignment statement

- 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



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



Cascaded assignment 串联.

- We can set multiple variables into the same value using a single assignment statement

Example

```
>>> z = y = x = 2 + 7 + 2
>>> x, y, z
(11, 11, 11)
```

or: print(x, y, z).

if: print(x, x, x)

(true/false)

整数、浮点数、复数、布尔型 → 11, 11, 11.

赋值：数字 字符串 列表，对象...

Practice

注意："apple"

☆：从右向左赋值.

x=1, y=2 分开两行，最后有 print..

- Write a program to exchange the values of two variables without using simultaneous assignment

S.a: x=1, y=2

x, y=y, x.

print(x, y).

方法：

z=x

x=y

y=z

print(x,y)

同时发生的.

Simultaneous assignment

- The values of two variables can be exchanged using simultaneous assignment

Example

```
>>> c = "deepSecret"           # Set current password.
>>> o = "you'll never guess"   # Set old password.
>>> c, o                       # See what passwords are.
('deepSecret', "you'll never guess")
>>> c, o = o, c                # Exchange the passwords.
```

Bad use of simultaneous assignment

一步步更好.

```
>>> # A bad use of simultaneous assignment.
>>> x, y = (45 + 34) / (21 - 4), 56 * 57 * 58 * 59
>>> x, y
(4.647058823529412, 10923024)
>>> # A better way to set the values of x and y.
>>> x = (45 + 34) / (21 - 4)
>>> y = 56 * 57 * 58 * 59
>>> x, y
(4.647058823529412, 10923024)
```

☆分开更好.

precise.

Order evaluation 评估

- When we put operators together, Python needs to know which one to do first
- This is called "operator precedence" 运算次序.
- Which operator "takes precedence" over the others 优

Example: X = 1 + 2*3 – 4 / 5 ** 6

X = 1 + 2*3 – 4 / 5 ** 6

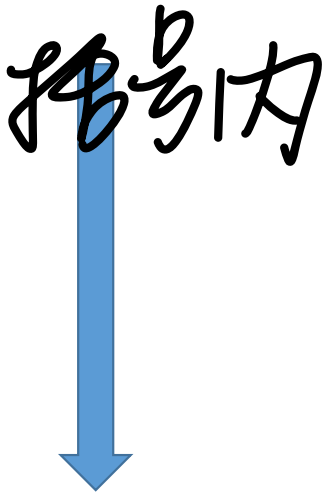
Numeric expression and operators

- We use some keys we have on the keyboard to denote the classic math operators
- Asterisk (*) is the multiplication operator
- Double asterisk (**) is used to denote Exponentiation (raise to a power)

Operator	Operation
+	Addition
-	Subtraction
*	Multiplication
/	Division
**	Power
%	Remainder

Operator precedence rules

- **Highest to lowest precedence rule**



✓ Parenthesis are always with highest priority

✓ Power

✓ Multiplication, division and remainder

✓ Addition and subtraction

✓ Left to right

Operator precedence

a=200, b=40 a/b=? ⇒ float, 5.0

a=25, b=6, a**b=? ⇒ float, 15.0

a=5, b=3, a**b=125 ⇒ int; a+b=8 ⇒ int

Example: x = 1+2**3/4*5

x**=3**2 ⇒ x=x·3^2.

x=1+2^3/4·5

x**=3+2 ⇒ x=x·(3+2)

整分数的后类型：

a=100+200

⇒ a=300

a='100'+'200'

⇒ a='100200'

a=3.6.

☆ round(x,y) (x为输入数，y为希望保留的位数)

⇒ a=int(a)=3.

直接取整，不四舍五入。

Floor division

```
>>> time = 257                  # Time in seconds.
>>> minutes = time // 60 # Number of complete minutes in time.
>>> print("There are", minutes, "complete minutes in", time, "seconds.")
There are 4 complete minutes in 257 seconds.
>>> 143 // 25
5
>>> 143.4 // 25
5.0
>>> 9 // 2.5
3.0
```

Floor division

-2 / 3 = ? -1.

- 整理：所有已学函数 a b.

x, y = divmod(a, b) 返回商与余数

float() int变float

str(a) a转为'a' 变字符串

sep=间隔.

end=结尾.

input("...")特殊操作：eval()直接运算.

divmod() 返回商和余数.

```
>>> time = 257          # Initialize time.
>>> SEC_PER_MIN = 60 # Use a "named constant" for 60.
>>> divmod(time, SEC_PER_MIN)   # See what divmod() returns.
(4, 17)
>>> # Use simulataneous assignment to obtain minutes and seconds.
>>> minutes, seconds = divmod(time, SEC_PER_MIN)
>>> # Attempt to display the minutes and seconds in "standard" form.
>>> print(minutes, ":", seconds)
4 : 17
>>> # Successful attempt to display time "standard" form.
>>> print(minutes, ":", seconds, sep="")
4:17
>>> # Obtain number of quarters and leftover change in 143 pennies.
>>> quarters, cents = divmod(143, 25)
>>> quarters, cents
(5, 18)
```

sep="xx"表示字符串间以什么作为间隔.

返回为元组.

tip: type(divmod(257, 60)[0]) ⇒ 整数类型

tuple

Augmented assignment 增量赋值

```
>>> x = 22
>>> x = x + 7
```

Augmented assignment

- The general form of augmented assignment looks like

```
<lvalue> <op>= <expression>
```

Example

```
>>> x = 22          # Initialize x to 22.
>>> x += 7          # Equivalent to: x = x + 7
>>> x
29
>>> x -= 2 * 7      # Equivalent to: x = x - (2 * 7)
>>> x
15
```

Personal tips

- Use parenthesis 括号.
- Keep mathematical expressions simple so that they are easy to understand
- Break up long series of math expressions to make them easy to understand

Integer division in Python

- In Python 2, Integer division truncates
- Integer division produces floating point numbers in Python 3
- The conversion between integer and floating point numbers is done automatically in Python 3
- Things change between Python 2 and Python 3

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)
```

300

字符串

字符串

"100200"

string

Type matters

- Python knows what type everything is
- Some operations are prohibited on certain types
- You cannot “add 1” to a string 字符串不可 乘除
- We can check the type of something using function type()

Types of numbers

- Numbers in Python generally have two types:
  - ✓ Integers: 1, 2, 100, -20394209
  - ✓ Floating point numbers: 2.5, 3.7, 11.32309, -30.999

- There are other number types, which are variations on float and integer (e.g., complex numbers)

★ a=40, b=60, a/b=? ⇒ float, 0.6

int(a) / int(60) ⇒ float, 0.6

Type can change

- The type of a variable can be dynamically changed
- A variable's type is determined by the value that is last assigned to the variable

```
>>> x = 7 * 3 * 2
>>> y = "is the answer to the ultimate question of life"
>>> print(x, y)          # Check what x and y are.
42 is the answer to the ultimate question of life
>>> x, y                 # Quicker way to check x and y.
(42, 'is the answer to the ultimate question of life')
>>> type(x), type(y)     # Check types of x and y.
(<class 'int'>, <class 'str'>)
>>> # Set x and y to new values.
>>> x = x + 3.14159
>>> y = 1232121321312312312312 * 98734237892374382 97
>>> print(x, y)          # Check what x and y are.
45.14159 121652559650716498712086836307354934126 64
>>> type(x), type(y)     # Check types of x and y.
(<class 'float'>, <class 'int'>)
```

adv. 动态地

△变量类型由最后一个值决定.

Type conversion

1.整数转换为浮点数
Float(1)
运行结果：1.0
Float(-1)
运行结果：-1.0
2.字符串转换为浮点数
Float('1.1')
运行结果：1.1
Float('-1.1')
运行结果：-1.1
3.布尔型转换为浮点数
Float(true)
运行结果：1.0
Float(false)
运行结果：0.0

- When an expression contains both integer and float, integers will be converted into float implicitly
- You can control this using functions int() and float()

string和int, float间转换：

b='6.6', b=int(b) ✗

b=float(b) → 6.6

⇒只有b='6'可以转换.

- Example:

```
>>> print(float(99)/100)

>>> i=42
>>> type(i)

>>> f=float(i)
>>> print(f)

>>> type(f)

>>> print(1+2*float(3)/4-5)
```

int

42.0

float

String conversions 字符串转换.

- You can also use int() and float() to convert strings into numbers
- You will get an error if the string contains characters other than numbers

str(1+10+100) = '111' ('1+10+100')X

str('1'+'10'+'100') = 11010

Converting numbers into string

- We can convert numbers into string using function str()

```
>>> str(5)                  # Convert int to a string.
'5'
>>> str(1 + 10 + 100)       # Convert int expression to a string.
'111'
>>> str(-12.34)             # Convert float to a string.
'-12.34'
>>> str("Hello World!")     # str() accepts string arguments.
'Hello World!'
>>> str(divmod(14, 9))      # Convert tuple to a string.
'(1, 5)'
>>> x = 42
>>> str(x)                  # Convert int variable to a string.
'42'
```

User input

- We can instruct Python to stop and take user inputs using function input()
- The input() function returns a string

Practice

- The BMI (body mass index) of a human can be calculated using the following equation:

$$BMI = weight\ (kg) \div height\ \hat{}\ 2\ (m)$$

- Write a program to input a user's weight and height, and then output his BMI

Converting user input

- If we want to read a number using input(), we must then convert the input into a number using int() or float()

- Later we will deal with bad input data

Comments

解释

- 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

String operations

- Some operators apply to strings

✓"+": concatenation

✓"*": multiple concatenation

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

Practice

- Write a program to instruct the user to input two of his friends' names, and then output a sentence "I am the friend of XX and XX."

Output using Print()

```
>>> print(42, "42") # An int and a str that looks like an int.
42 42
>>> print('3.14')    # A str that looks like a float.
3.14
>>> print(3.14)     # A float.
3.14
```

① print(a, b, c) ⇒ 占据一行.

② print(a) ⇒ a 占据两行.
print(b) b

有: a b c
无: a bc

③ print(a, b, end='') '' 有无空格分隔.

print(c)

④ print(a,b,c, sep='') '有无空格不同

More details on print()

```
print(...)
    print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)

    Prints the values to a stream, or to sys.stdout by default.
    Optional keyword arguments:
    file:  a file-like object (stream); defaults to the current sys.stdout.
    sep:   string inserted between values, default a space.
    end:   string appended after the last value, default a newline.
    flush: whether to forcibly flush the stream.
```

Examples

```
>>> print("I", "like", "CS")
I like CS
>>> print("I", "like", "CS", sep='')
IlikeCS
>>> print("I", "like", "CS", sep=',')
I,like,CS
```

Example

```
print("Test line 1")
print("Test line 2")

print("Test line 1",end = " ")
print("Test line 2")

print("Test line 1",end = "---")
print("Test line 2")
```

主要用来实现Python中各种数据类型

A powerful function - eval()

与str之间的转换. 将字符串转成表达式并执行.

- The eval() function takes a string argument and evaluates that string as a Python expression, i.e., just as if the programmer had directly entered the expression as code

- The function returns the result of that expression .

- Eval() gives the programmers the flexibility to determine what to execute at run-time

- one should be cautious about using it in situations where users could potentially cause problems with “inappropriate” input

→ print 结果不带引号.

Example

```
>>> string = "5 + 12" # Create a string.
>>> print(string)     # Print the string.
5 + 12
>>> eval(string)      # Evaluate the string.
17
>>> print(string, "=", eval(string))
5 + 12 = 17
>>> eval("print('Hello World!')") # Can call functions from eval().
Hello World!
>>> # Using eval() we can accept all kinds of input...
>>> age = eval(input("Enter your age: "))
Enter your age: 57.5
>>> age
57.5
>>> age = eval(input("Enter your age: "))
Enter your age: 57
>>> age
57
>>> age = eval(input("Enter your age: "))
Enter your age: 40 + 17 + 0.5
>>> age
57.5
```

字符串.

直接计算✓.

不要漏括号.

否则报错.

Example

```
>>> eval("10, 32")          # String with comma-separated values.
(10, 32)
>>> x, y = eval("10, 20 + 12") # Use simultaneous assignment.
>>> x, y
(10, 32)
>>> # Prompt for multiple values.  Must separate values with a comma.
>>> x, y = eval(input("Enter x and y: "))
Enter x and y: 5 * 2, 32
>>> x, y
(10, 32)
```

Other functions

How to get a substring from a string?

Fifteen Essential Rules for University E-mail Communication

The 1st and most essential rule is: Always write your English professors in English.

Do not write in Chinese. This is an *English* class. Your western professors probably can't read Chinese and you need the practice in English, anyway!

Rule #2:
Fill in the subject line with a very few factual words. Professors and administrators receive lots of email. Weird subject lines will get your email delayed, if not outright deleted.

For example:

Missing class today.
Question about the assignment.
Problem with Blackboard.
Requesting an appointment.
Question about the online quiz.
A grammar question.
Apology for late submission.
Help with an application.
A question about policy.

Please DO NOT offer lines like:
DoNt miss ReaDiNg this LOL.
The funnest story you will 4ever read!!!!!
I’m such a idiot! 
Hi
Can you help me with this?
just wondering
you ROCK!!!!!!!!!!
Confuzion
tha reel deel

Rule #3:
Begin your email by introducing yourself with your full name in pinyin. End your email with your personal name in pinyin.

Rule #4:
Tell your professor your section number and class time.

Rule #5:

If you are writing an administrator, you should include your student ID number and your year at CUHKSZ (eg: freshman, sophomore, junior, senior, graduate student). You may also need to tell administrators your college (eg: Shaw, Diligentia) and/or your school (eg: HSS, SME, SSE).

For example:
Good morning, professor, this is Tan Twan Eng from your 8:30 class on Monday, section 36.

For example:
Good morning. My name is Twan Eng and I am a student from Diligentia College and SME. My student ID number is 020200000.

Rule #6:
Get to the point.
Professors and administrators are busy and receive many emails. Your message must be quickly understood. Keep your message short. Five or six sentences at most. If you MUST write a longer message, be as brief as possible. Don’t wander off topic. Don’t philosophize. If you want to have a long conversation, request a face-to-face meeting.

For example:

Good morning. My name is Twan Eng and I am a student from Diligentia College and SME. My student ID number is 020200000. I am writing to request a meeting with an academic advisor to discuss changing my major.

Rule #7:
Be polite!
The key word here is to manage your tone. Be polite. You may be frustrated, or even angry. But do not let your frustration show in your message. You don’t want to offend anyone or start a fight. If you want to solve a problem, then keep calm. Explain the problem and work together to build a solution!

Rule #8:
Attachments.
If your email includes an attachment, please say so and identify the attached file or files. Then be certain that you have attached the correct file or files!

For example:
I have attached a copy of my medical report and a letter from my physician.

I have attached the application and a copy of my receipt for payment.

I have attached my transcript and a letter of recommendation from Professor X.

Rule #9:
Use your best English.
Remember that all writing is practice for your final exam. Write in complete sentences and proofread your email before you hit “send.” You need the practice, and your reader wants to understand your email with no confusion!

Rule #10:
When you send email to people within the university, use your official university email account.

Rule #11:
Avoid “txt spk.”
In other words, don’t use “text speech” or other shortcuts. Please write your message in the best English that you can. Avoid “slang” text speech like “thx” and “wtf.” You are now living and working in a professional environment. You are learning how to communicate like professionals.

Rule #12:
Please don’t use emojis.
University professionals just need the facts. We have a lot of mail to read and we make decisions based upon contents, not decorations.

And under no circumstances should you ever use a “lewd” emoji in professional communication!

Rule #13:
Conclude with a polite “sign off.”
You’ll want your closing to be friendly, professional, and respectful. You may also want to reflect your personality and to acknowledge your good relationship with your professor.

For example:

Common endings include:

Thank you very much,

Thanks for your time,

Sincerely,

All the best,

Thanks so much,

Thank you,

With appreciation,

With thanks,

Respectfully,

Regards,

Thanks,

With regards,

Your student,

With thanks,

Best wishes,

Sincerely yours,

Avoid unprofessional endings like:

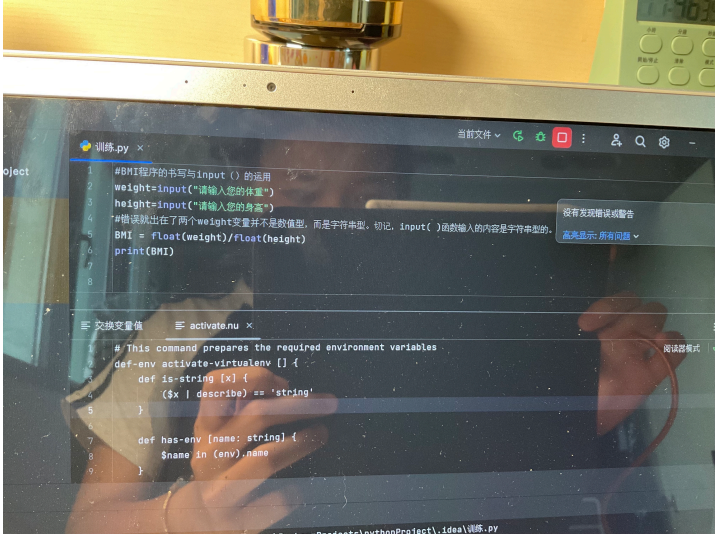
Sent from my i-phone,
Awaiting your reply,
Love,
Yours truly,
Expectantly,
Urgent,
Hopefully,
Faithfully,
Always,
In anticipation,
Thx,
Top priority,

Rule #14

If your email is an emergency, and you need a call back, say so! Then be sure to leave your mobile number. But only ask for call-backs in real emergencies.

Rule #15

Before you hit “reply” make certain that you have proofread your email, that your message is polite, and that you have included all the necessary information including contact details. Then. Do not hit “reply to all” unless you must send everyone in the chain a copy of the email. Otherwise, just hit “reply.”


训练.py
当前文件
#BMI程序的书写与input（）的运用
weight=input("请输入您的体重")
height=input("请输入您的身高")
#错误就出在了两个weight变量并不是数值型，而是字符串型。切记，input（）函数输入的内容是字符串型的。
BMI = float(weight)/float(height)
print(BMI)
没有发现错误或警告
高亮显示: 所有问题
交换变量值
activate.nu
# This command prepares the required environment variables
def-env activate-virtualenv [] {
def is-string [x] {
($x | describe) == 'string'
}
def has-env [name: string] {
$name in (env).name
}
阅读器模式