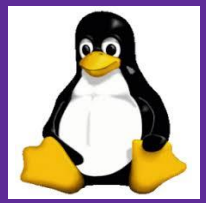
Introduction to Computation

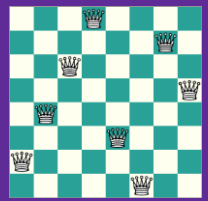
Autumn, 2023

Prof. Fan Cheng

Shanghai Jiao Tong University

chengfan85@gmail.com
https://github.com/ichengfan/itc





2

Outline

- Character
- Number
- String
- Variable

Structure of a Program

```
answer = 100

guess = int(input("Please enter a number between 1 and 200: "))

while guess != answer:

    if guess > answer:
        print("Your guess is too large")
    else:
        print("Your guess is too small")

    guess = int(input("Please enter a number between 1 and 200: "))


print("Congratulations! Your guess is correct.")
```

基本语法

- 输入、输出
- 变量定义
- 条件判断语句
- 循环
- 函数

Learn English:

character, word, sentence, passage, section, chapter

字词句段篇章

Character

Bit

- 在计算机中，所有的数据都是用二进制表示的: 高电压表示1，低电压表示0
- 用bit来表示1位数据，也就是0或者1
- 用8个bit表示1个byte；1024个byte表示1KB，1024KB表示1MB，1024MB表示1GB，1024GB表示1TB
  - 更大的TB, PB, EB, ZB
- CPU读取指令，然后执行。指令的长度，可以为4, 8, 16, 32, 64等等
  - 以4位CPU为例，用长度为4位的二进制表示数据 $(1011)_2, (0011)_2$等等
  - n为CPU的指令长度，最多表示大小$0 - 2^n$的数据，超过会溢出
- 按位取反（补码）： 对整数的每位做取反操作， $0 \rightarrow 1, 1 \rightarrow 0$
  - 以8位整数为例：$11001100 \rightarrow 00110011$
  - 假设整数为 x, 取反操作用$\sim x$ 表示: ~7=-8, ~1= -2
- $x + \sim x + 1 = 0$
  - $x + \sim x = (111 \ldots 111)_2$
  - $(111 \ldots 111)_2 + 1 = 0$ (超过最大长度，溢出)
  - $(111 \ldots 111)_2 = -1$

```
5   print(1, ~1)
6   print(12, ~12)
7   print(123, ~123)
8   print(1234, ~1234)
9   print(0, ~0)
10  print(-1, ~-1)
11  print(-12, ~-12)
12  print(-123, ~-123)
13  print(-1234, ~-1234)
```

```
1 -2
12 -13
123 -124
1234 -1235
0 -1
-1 0
-12 11
-123 122
-1234 1233
```

Characters in computers

- Digits: 0-9
- Letters: a-z, A-Z
- +, -, *, /
- (), [], {}, !, “,”, ‘,’,
- $, @

In standard keyboard, 101-107 keys

- Symbols in ‘’ or “” are characters: ‘a’, ‘b’, ‘1’, ‘*’, “a”, “c”, “&”
  - ‘1’ is different from 1.
- Computers know only 0 and 1. All the characters are stored in 0’s and 1’s .
- For example,
  - ‘a’ is denoted by 1100001
  - ‘1’ is denoted by 110001
- To exchange information between computers, a standard is necessary. ASCII

ASCII

ASCII, abbreviated from American Standard Code for Information Interchange, is a character encoding standard for electronic communication.

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

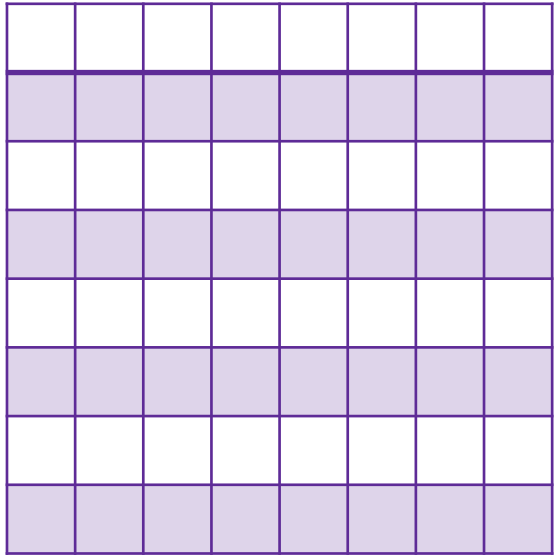
Source: www.asciitable.com

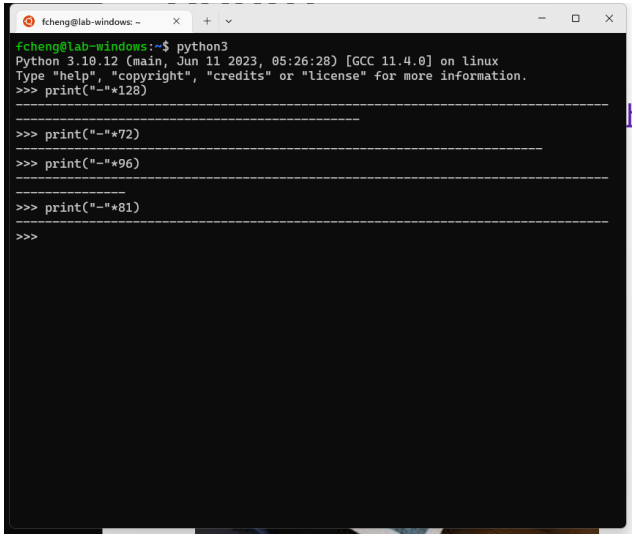
- ASCII codes represent text in computers, telecommunications equipment, and other devices.
- Most modern character-encoding schemes are based on ASCII, although they support many additional characters.
- Dec: 10进制
- Hx：16进制
- Oct：8进制

Remember some frequently used characters; e.g., ‘a’, ‘A’, ‘0’
Keep in mind: ‘a’—’z’, ‘A’—’Z’, ’0’—’9’ are continuous.

print()

- print在字面上：打印
- 从打印的角度理解：光标在纸面上移动，从左往右，一行一行
- 换行: enter
- 回退: backspace
- print(123), print('123')效果一样





Escape character

- 字母、数字，数学运算符号都是直接表示：’a’, ‘Z’, ‘0’, ‘9’, ‘+’, ‘-’, ‘*’, ‘/’
- 如何将键盘上一些特殊的键位写成字符？ 例如：回车、换行、制表符
- 思路：用两个字符的组合来表示，以\(反斜线)开始

  \n: new line, \t: tab, \\: \, \a: bell print(‘\a’)
- Characters with special meaning, called escape characters (转义字符)
- 特别注意：当一个字符以\ 开始的时候，要注意这不是普通字符，而是转义字符，要结合后面的字符一起判断。否则是语法错误

Escape Sequence	Meaning
newline	Ignored
\\	Backslash (\)
\'	Single quote (')
\"	Double quote (")
\a	ASCII Bell (BEL)
\b	ASCII Backspace (BS)
\f	ASCII Formfeed (FF)
\n	ASCII Linefeed (LF)
\r	ASCII Carriage Return (CR)
\t	ASCII Horizontal Tab (TAB)
\v	ASCII Vertical Tab (VT)
ooo	ASCII character with octal value *ooo*
\x*hh...*	ASCII character with hex value *hh...*

- \\：表示反斜线\
- \r：回到当前行的开始(首字符)
- \b：backspace 光标回退一位
- \a：铃声（控制台）
- \n：换行，另起一行
- \t：tab键
- \f： 光标换到下一行的相同位置
- \和/是不同的

```
15  print('abc\\123')
16  print('abc\r123')
17  print('abc\b123')
18  print('my bell \a')
19  print('abc\n123')
20  print('abc\t123')
21  print('abc\f123')
```

```
abc\123
123
ab123
my bell
abc
123
abc     123
abc
   123
```

Character and its ASCII

字符和其对应的二进制表示的相互转换

- ord() and chr() : Two useful function for dealing with ASCII.
- ord(x): return the corresponding ASCII of character x
- chr(x): return the corresponding character by its ASCII

```
print(ord('a'))
print(ord('A'))
print(ord('z'))
print(ord('Z'))
print(ord('$'))
print(ord('!'))
print(ord('+'))
print(ord('-'))
print(ord('('))
print(ord(')'))
print(ord('1'))
print(ord('0'))
print(ord('9'))
```

```
97
65
122
90
36
33
43
45
40
41
49
48
57
```

```
print(chr(97))
print(chr(65))
print(chr(122))
print(chr(90))
print(chr(36))
print(chr(33))
print(chr(43))
print(chr(45))
print(chr(40))
print(chr(41))
print(chr(49))
print(chr(48))
print(chr(57))
```

```
a
A
z
Z
$
!
+
-
(
)
1
0
9
```

```
print(ord('\n'))
print(ord('\f'))
print(ord('\b'))

print(chr(10))
print(chr(12))
print(chr(8))
```

```
10
12
8
```

- Functions (函数): print(), chr(), ord()
- Only 255 characters in ASCII. No 中文
- Unicode is more widely used in practice.
- ASCII is a subset of Unicode

Number

Python data types

Python数据类型：物以类聚、人以群分

- Before, we have used numbers and words like 123+456, “Shanghai”
- In Python, we have two basic data types: **number** and **string**
  - Numbers can be **integer, float,** or **Complex number**
  - Integer can be **binary, octal** and **Hexadecimal**

Numbers

- 数学中，区分整数和实数（小数）。实数分为有理数、无理数
- 计算机，区分为整数和浮点数（float，实数）
- 计算机存储空间有限，不可能处理无限长度的数据
  - 整数是完整存储在计算机中
  - 浮点数是近似保存在计算机中，浮点数是不准确的，精度有限制
- 数字的写法和数学中一样，124, 456, 7.89, -3.14
- 科学计数法：aeb $:= a \times 10^b$
  - $1e3 = 10^3, 1.2e-5 = 1.2 \times 10^{-5}$
  - e的前后没有空格
- 数字的使用和数学中一样，四则混合运算 +， -， *， /
  - 123 + 456, 123 - 456.0, 123 / 333
  - 123 * 456 - 789
- 幂 **：2**10=1024, 3**3 = 27
  - 两个**连在一起，中间没有空格
- 和数学中一样，运算符有优先级，** 高于 *，/；*, / 高于+，-
- 和数学中一样，括号()可以改变运算顺序

```python
print(123, 456, -1, -2, 0)

print(123.00, 45.6, -8.9, -1.2)

print(1e-3, 1e4, 200e-1)

print(1+2, 2-5, 4/5, 4*8, 2**10)

print(2**100, 3**10, 3**0.5, 2**-1)

print(1+2/3+4, (1+2)/3+4)
```

```
123 456 -1 -2 0
123.0 45.6 -8.9 -1.2
0.001 10000.0 20.0
3 -3 0.8 32 1024
1267650600228229401496703205376 59049 1.7320508075688772 0.5
5.666666666666666 5.0
```

```python
print(1e 3)

print(3* * 3)
```

```
  File "c:/Users/popeC/OneDrive/CS124计算导论/2021 秋季/2021.py", line 13
    print(1e 3)
          ^
SyntaxError: invalid syntax
```

```
  File "c:/Users/popeC/OneDrive/CS124计算导论/2021 秋季/2021.py", line 15
    print(3* * 3)
            ^
SyntaxError: invalid syntax
```

必须严格按照语法

Integer

Python中，整数的长度是没有限制的，既可以表示任意精度的整数(不会溢出)

- In default, integers are based on 10. For example, 96.
- Binary(二进制): 0b (0B) 96 = 0b1100000 or 0B1100000
- Octal(八进制): 0o (0O) 96 = 0o140 or 0O140
- Hexadecimal(16进制): 0x (0X) 96 = 0x60 or 0X60.
  - 10,11,…,15 are 'ABCDEF' or 'abcdef'
  - 8 = 0x8, 9 = 0x9, 10= 0xA or 0xa, …, 15 = 0xF or 0xf.
- Underscores in Numeric Literals (Python 3.6)
  - 123,456,789(English countries) 123_456_789
  - 0xFFF_FFF_FAA, 123.456_789
- How to transform a number from binary to oct or hex? 3 by 3! 4 by 4!

  127 = 0b1 111 111 = 0o177

  = 0b 111 1111 = 0x 7F

```
x = 96
print(x)
```

```
96
```

```
x = 0b1100000
print(x)

x = 0B1100000
print(x)
```

```
96
96
```

```
x = 0o140
print(x)

x= 0O140
print(x)
```

```
96
96
```

```
x = 0x60
print(x)

x= 0X60
print(x)
```

```
96
96
```

```
print(0b1111111,0B1111111, 0o177, 0O177, 0x7F, 0X7F)
```

```
127 127 127 127 127 127
```

```
56   print(123_456_789, -123_7_8_9)
57   print(0xFFF_FFF_FAA, 0b111_000_111)
```

```
123456789 -123789
68719476650 455
```

```
23   print(2**100)
24   print(2**300)
```

```
1267650600228229401496703205376
203703597633448608626844568840937816105146839366593625063614044935438129976333670618339737 6
```

Floating-Point Number (浮点数)

- In mathematics, we have **rational number** and **irrational number**
- In python, we call it floating-point number (浮点数，小数)
  - 4.2, 3.1415926, 10000.431
  - 1e-8, 3e4 (1 and 3 cannot be missed)
- Floating numbers are not accurate. Integers are always accurate. Rounding error!
- ***Complex number*** are denoted by a + b j, a is the real part and b is the imaginary part.
  - When b=1, 1 cannot be missed

```
27  print(1/3)
28  print(1e3)
29  print(1e5)
30  print(1e-6)
31  print(0.00001)
32  print(3.14e5)
33  print(3.14e-5)
34  print(-1e-4)
35  print(-1e4)
36  print(-2.1e4)
37  print(-2.1e-4)
38  print(3.14e20)
39  print(3.14e-20)
```

```
0.3333333333333333
1000.0
100000.0
1e-06
1e-05
314000.0
3.14e-05
-0.0001
-10000.0
-21000.0
-0.00021
3.14e+20
3.14e-20
```

```
41  print(1+2j)
42  print(1+1j)
```

```
(1+2j)
(1+1j)
```

```
>>> print(1e4)
10000.0
>>> print(e4)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'e4' is not defined
```

```
>>> print(1+2j)
(1+2j)
>>> print(1+1j)
(1+1j)
>>> print(1+j)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'j' is not defined
```

Floating Accuracy (浮点数不准确)

- 尽量避免使用浮点数，尽可能使用整数

```
12   print(.1)
13   print(.1 + .1 + .1)
```

```
0.1
0.30000000000000004
```

```
26   print(-19999999999999999999)
27   print(1234556799999999999999)
28   print(3.1111111119999999)
29   print(3.00000000000000000001)
```

```
-19999999999999999999
1234556799999999999999
3.111111112
3.0
```

- 能够用整数的地方一定要用整数

```
46   x = 13.949999999999999999
47   print(x)
```

```
13.95
```

整数和浮点数是两种不同的类型

https://docs.python.org/3/tutorial/floatingpoint.html

Expressions on numbers

- In python, numbers and operators can form expression; e.g., 3+4, 2/5, etc.
- Mathematical operators: +, -, *, /, //, **, %
  - 3/4, 3//4, 2**100, 100%21
  - /(数学除法): floating-point division
  - //(整除): integer division, keep the integer part after mathematical division.
  - a**b: a^b. (注意：**没用空格)
  - %: modulus(模、同余、余数), (%) yields the remainder after integer division

```
print(3/4)
print(3.0/4)
print(3/4.0)
print(3.0/4.0)

print(3//4)
print(30//4)
print(-3//4)
print(40//4)
print(40/4)
```

```
0.75
0.75
0.75
0.75
0
7
-1
10
10.0
```

```
print(-3%3)
print(-1%3)
print(-2%3)
print(0%3)
print(1%3)
print(2%3)
print(3%3)
print(4%3)
print(5%3)
print(6%3)
print(7%3)
```

```
0
2
1
0
1
2
0
1
2
0
1
```

```
print(10-30//4, -30//4)
```

```
3 -8
```

常见错误:

$\frac{b}{2ac}$写成$b/2*a*c$

$x^{0.5}$ 写成 $x**1/2$

优先级！！！

Logic expression (逻辑表达式)

- In logic, if the expression is satisfied, then we call it **True**; otherwise we call it **False**
- In python, True is **1** and False is **0**
- Logic operators: >, <, ==, >=, <=, !=
  - 3>2, 1<-4, 7>=100, 45<=53
  - >, <, >=, <= are just the normal mathematical relation
  - ==: x==y, x is equal to y
  - !=: x!=y, x is not equal to y

```
print(1+2+3 > 5)
print(1+2+3 < 5)
print(1+2+3 == 5)
print(1+2+3 != 5)
print(5 == 5)
print(5 > 5)
print(5 < 5)
print(5 >= 5)
print(5 <= 5)
```

```
True
False
False
True
True
False
False
True
True
```

```
print(True == 1)
print(False == 1)
print(True == 0)
print(False == 0)
```

```
True
False
False
True
```

Q: True * 12?

Remainder 余数

计算 $123456789^{987654321}$除以2022的余数

- 定义：$x = a\ (mod\ n)$表示x同余a，除数(模)是n
- 例子 $7 = 3\ (mod\ 4), 9 = 1\ (mod\ 2), 1234567 = 25\ (mod\ 111)$

  $7 = -1(mod\ 4), 8 = -2\ (mod\ 10)$
- 基本规律 $a = b, c = d\ (mod\ n)$, 那么
  - $a + b\ =\ c + d, a - b\ =\ c - d, a * b\ =\ c * d$
- 判断两个数同余，不能用$a == b$, 要用$(a - b)\%n == 0$
- 在计算大整数的余数的时候，用同余运算来处理，计算量更小

  $1234567 = 25\ (mod\ 111)$

  $1234567 * 1234567 = 25 * 25 = 70\ (mod\ 111)$

  不用直接计算 $1234567 * 1234567$的值
- 例子：$1234567654321\ **\ 100\ =\ 1\ **\ 100\ =\ 1\ (mod\ 111)$

String

String

- A **string** is a sequence of characters, enclosed by **single quotes** and **double quotes**
  - Single quotes and double quotes are **identical**
  - ‘abc’ ==“abc”, “SJTU” ==‘SJTU’

```
print("I am Chinese")
print("I\n am \n Chinese")
print("\t I \t am \t Chinese")
```

```
I am Chinese
I
 am
 Chinese
    I   am      Chinese
```

```
print('I am Chinese')
print('I\n am \n Chinese')
print('\t I \t am \t Chinese')
```

```
I am Chinese
I
 am
 Chinese
    I   am      Chinese
```

```
print("上海交通大学")
print('上海交通大学')
```

```
上海交通大学
上海交通大学
```

注意：在string中，’或’’会和前面最近的一个’或’’匹配

```
print("""")
```

```
    ^
SyntaxError: EOF while scanning triple-quoted string literal
```

String: triple quotes

- String literals can span multiple lines. One way is using triple-quotes: """..."""" or '''...'''. End of lines are automatically included in the string, but it's possible to prevent this by adding a \ at the end of the line.
- Triple quotes are useful for large chunks of text in program code spreading several lines.

```
print("""I am Chinese""")
print('''I am Chinese''')
print("I am Chinese")
print('I am Chinese')
```

```
I am Chinese
I am Chinese
I am Chinese
I am Chinese
```

```
print("""《赴戍登程口占示家人》
力微任重久神疲，再竭衰庸定不支。
苟利国家生死以，岂因祸福避趋之？
谪居正是君恩厚，养拙刚于戍卒宜。
戏与山妻谈故事，试吟断送老头皮。
""")
print("""《赴戍登程口占示家人》\
力微任重久神疲，再竭衰庸定不支。\
苟利国家生死以，岂因祸福避趋之？\
谪居正是君恩厚，养拙刚于戍卒宜。\
戏与山妻谈故事，试吟断送老头皮。\
""")
```

“…”与‘…’不能跨行。
要跨行必须在每行末尾用\

```
《赴戍登程口占示家人》
力微任重久神疲，再竭衰庸定不支。
苟利国家生死以，岂因祸福避趋之？
谪居正是君恩厚，养拙刚于戍卒宜。
戏与山妻谈故事，试吟断送老头皮。

《赴戍登程口占示家人》力微任重久神疲，再竭衰庸定不支。苟利国家生死以，岂因祸福避趋之？谪居正是君恩厚，养拙刚于戍卒宜。戏与山妻谈故事，试吟断送老头皮。
```

String with Escape character

```
print("I \t am \t from \t Shanghai Jiao Tong University. \nI love this place")
```

```
I	am	from	Shanghai Jiao Tong University.
I love this place
```

Escape Sequence	Meaning
newline	Ignored
\\	Backslash (\)
\'	Single quote (')
\"	Double quote (")
\a	ASCII Bell (BEL)
\b	ASCII Backspace (BS)
\f	ASCII Formfeed (FF)
\n	ASCII Linefeed (LF)
\r	ASCII Carriage Return (CR)
\t	ASCII Horizontal Tab (TAB)
\v	ASCII Vertical Tab (VT)
ooo	ASCII character with octal value *ooo*
\x*hh...*	ASCII character with hex value *hh...*

- \出现的时候就是转义字符
- 系统会把\后面的几个字符和\合起来理解，而不是单独考虑
  - \", \' 都是转移字符
  - \r: 回到当前行的开始
  - \b: backspace 光标回退一位
  - \a：铃声 （在控制台下测试）
  - \n: 换行，另起一行
  - \f: 光标换到下一行的相同位置
- 记住：\n,\b,\t

```
print("This is a great world.\n Welcome here.")
print("This is a great world.\t Welcome here.")
print("This is a great world.\b Welcome here.")
print("This is a great world.\r Welcome here.")
print("This is a great world.\f Welcome here.")
```

```
This is a great world.
 Welcome here.
This is a great world.	 Welcome here.
This is a great world Welcome here.
 Welcome here.t world.
This is a great world.
                      Welcome here.
```

String with escape characters

- “ and ‘ have been used by system. What should we do if we want to print “ or ‘?

```
print("""")
```

```
  File "C:\Users\fcheng\Desktop\test3.py", line 2
    print("""")
            ^
SyntaxError: EOF while scanning triple-quoted string literal
```

```
print('''')
```

```
  File "C:\Users\fcheng\Desktop\test3.py", line 2
    print('''')
            ^
SyntaxError: EOF while scanning triple-quoted string literal
```

```
print("\"")
print("\'")
```

```
"
'
```

- How to output \: print(‘\\’)

```
print("\")
```

语法错误

```
print("\\")
print('123456789\b\b\b\b-.-')
```

```
\
12345-.-9
```

思考

```
print('\')
print('\\')
print('\\\')
print('\\\\')
```

String with escape characters

- Python中如何**取消转义字符**?

  Add “r” before string can be used to stop escape character.

```
print(r"I \t am \t from \t Shanghai Jiao Tong University. \nI love this place")
```

```
I \t am \t from \t Shanghai Jiao Tong University. \nI love this place
```

```
I    am      from    Shanghai Jiao Tong University.
I love this place
```

重要的事情说三遍

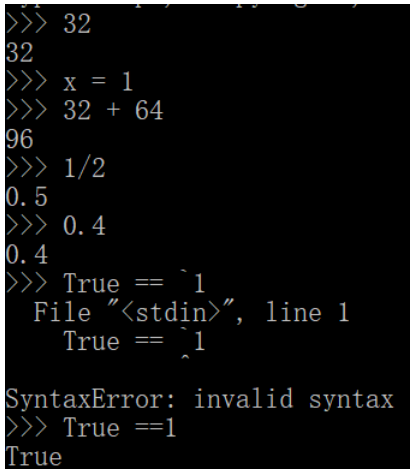
Escape character

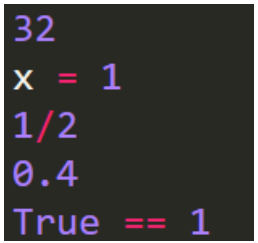
escape character

escape character

Python console and IDE

- In the console (interpreter) that the python 3.x package attached, when you type sth. in the console, you will get an output after you type newline, even if you don't use print(). However, in python IDE, like PyCharm, that is not the same. In PyCharm, you won't get an output unless you use print()





In VsCode/PyCharm, you won't get any output since you don't use print().

- The reason is, Python console is designed to help you learn python grammars. It will automatically output the value of the input for convenience. This behavior is not defined by Python Language and it is controlled by Python console. **PyCharm/VS Code will operate loyally as python Language defined**

Variable

Types of Storage

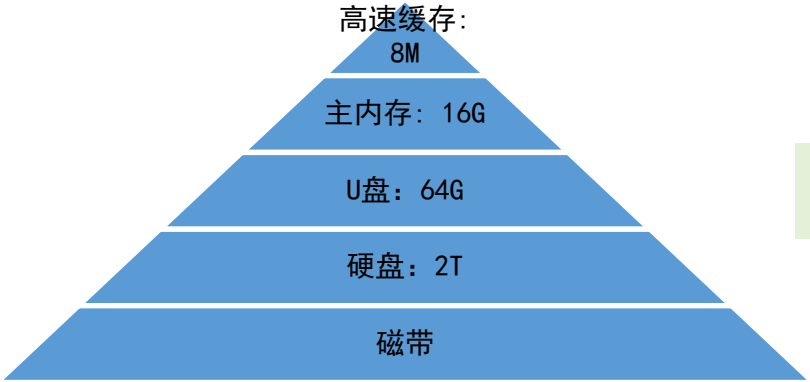


- Memory is primarily of three types –
  - Cache Memory (faster speed, limited size and expensive price)
  - Primary Memory/Main Memory **(↓,↑,↓)**
  - Secondary Memory (like hard-disk) **(↓,↑,↓)**

速度越来越快

价格越来越便宜

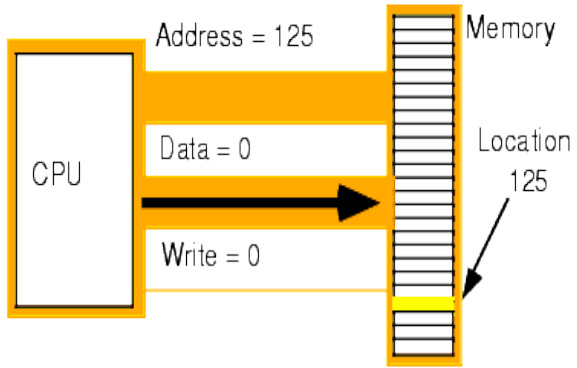
容量越来越大



- ✓ The faster, the more expensive
- ✓ The cheaper, the larger

Memory (内存)

- A memory is just like a human brain: **store data and instructions**.
- Computer memory is the storage space in the computer, where data is to be processed and instructions required for processing are stored
- The memory is divided into large number of small parts called **cells**
  - **Each location or cell** has a **unique address**



Some cells are readable and some are writeable.

https://www.tutorialspoint.com/computer_fundamentals/computer_memory.htm

Memory Abstraction (内存想象示意图)

	苟	利	国	家	生	死	以
		1	2	3			
			+	-			
		S	J	T	U		

- Memory are divided into cells, each of which has a unique address.
  - Each cell : unique address
- 用一个大方格（作文本）来表示一个内存
- 每个小方格可以用第i行，第j列作为它的地址 (i,j)
- 每个小方格可以存储一个信息
- 每个小方格的信息可以根据地址来访问：读 或者 写
- 后面写的信息会把前面写的信息覆盖掉

数据类型迥异，其地址的结构是一样的

Variable in python

- In mathematics, we use variables to conduct abstract analysis. $z^2 = x^2 + y^2$
- In computer programs, constants like ‘a’, ‘xyz’, 1, 2, are stored in memory.
  - We define variables to operates these constants
- **A variable is a name that refers to a value**
- Name of a variable can contain **letters, numbers and underscore**, but they have to begin with a letter or underscore character (_)
  - X, y , z, x, Y, z, price, cake, _step, x1, x2
  - 非法的: 2x, 3y, x+y, x@y, !x
- Variable names (also called identifier) can be arbitrarily long
- Variable names are **case-sensitive(大小写敏感)**, so spam, Spam, sPam, and SPAM are all different. Although it is legal to use uppercase letters, by convention we don’t
- For the most part, programmers are free to choose any name that conforms to the above rules. Good programmers always try to choose names that describe the thing being named (meaningful), such as message, price_of_tea_in_china.

Variable name: Keywords

- **Keywords** (also called reserved words) define the language's rules and structure, and they cannot be used as variable names. Python has thirty-five keywords:

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

- 用关键词作为变量名会报错
- True，False是关键词

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

Use a variable

- Variables **are created** when they are assigned (=)
  - year = 2018 (**assignment statement, 赋值语句**)
  - month = “September”
  - price = 3.81
- The type of the variable is determined by Python
  - year = 2018 year is int
  - month = “September” month is string
  - price = 3.81 price is float
- A variable can be reassigned (重新赋值) to whatever, whenever
  - price = 3.4 price is float
  - price = 2 price is int
  - price = “hello world” price is string
  - my_price = “hello world” my_price and price are the same

特别注意

1. 尽量用英文来定义变量，不要用拼音和汉字

2. 不要和系统抢变量名**,** 譬如**print**

3. 文件名也要用变量名的规则来定义**:** 字母、数字、下划线，**xxx.py**

Variable Example

```
x = 3
print(x)
y = 4.0
print(y)
z = -5.0
print(z)
my_name = "Python"
print(my_name)
_year = "2018"
print(_year)
my_Price = 34.56
print(my_Price)
e = 2.71828
print(e)
pi = 3.1415926
print(pi)
```

```
3
4.0
-5.0
Python
2018
34.56
2.71828
3.1415926
```

```
>>> $y = 100
  File "<stdin>", line 1
    $y = 100
    ^

SyntaxError: invalid syntax
>>> 9y = 100
  File "<stdin>", line 1
    9y = 100
     ^

SyntaxError: invalid syntax
>>> y = 100
>>> print(Y)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'Y' is not defined
>>> name = "SJTU2018"
>>> print(name)
SJTU2018
>>> print(nAme)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'nAme' is not defined
>>> while = 1
  File "<stdin>", line 1
    while = 1
          ^

SyntaxError: invalid syntax
>>> if = 1
  File "<stdin>", line 1
    if = 1
       ^

SyntaxError: invalid syntax
```

_, digitals, letters and keywords

type() and id()

- **type(x)**: will return the type of variable x
  - type(123)
  - type(12.3)
  - type(“Hello world”)

```
print(type(124))
print(type(12.34))
print(type(1+2j))
print(type("Hello world"))
print(type("SJTU 2018 IEEE"))
```

```
<class 'int'>
<class 'float'>
<class 'complex'>
<class 'str'>
<class 'str'>
```

- **id(x)**: will return the address of variable x

```
x = 1
y = 2.0
z = 1 + 4j
print(id(x), id(y), id(z))
print(id(1), id(2.0), id(1+4j))
```

```
498755712 49005520 48272592
498755712 49005520 47535648
```

```
name = "SJTU"
poem = "苟利"
print(id(name),id(poem))
print(id("SJTU"),id("苟利"))
```

```
47670144 47664688
47670144 47664688
```

id是由系统决定的，结果可能不同

Variable expressions

Python中基本的表达式和数学中一样

```
x = 1
y = 2
z = 3
w = 5
print(x+y)
print(x-y)
print(x*y*z)
print(x+y-z*w)
print(y**z)
print(z/y)
print(z//y)
print(w%y)
print(w%z)

pi = 3.1415926
r = 2.0
print(pi*r**2)
```

```
3
-1
6
-12
8
1.5
1
1
2
12.5663704
```

```
x = 1
y = 2
z = 3
w = 5

print(x+y>z)
print(y+z>x)
print(z+x>y)
```

```
False
True
True
```

- **Priority of operators**: *, /, // are higher than +, -. () is the highest

```
x = 1
y = 2
z = 3

print(x+y*z)
print((x+y)*z)
```

```
7
9
```

Use “()” if you are unsure of the order

最常见的逻辑错误

Common Errors

- Python中语句都要顶格写，开头不能任意加空格

```
103    print("Hello world")
104     print("Hell world")
```

```
103    x = 1
104     y = 2
```

- 转变数学思维

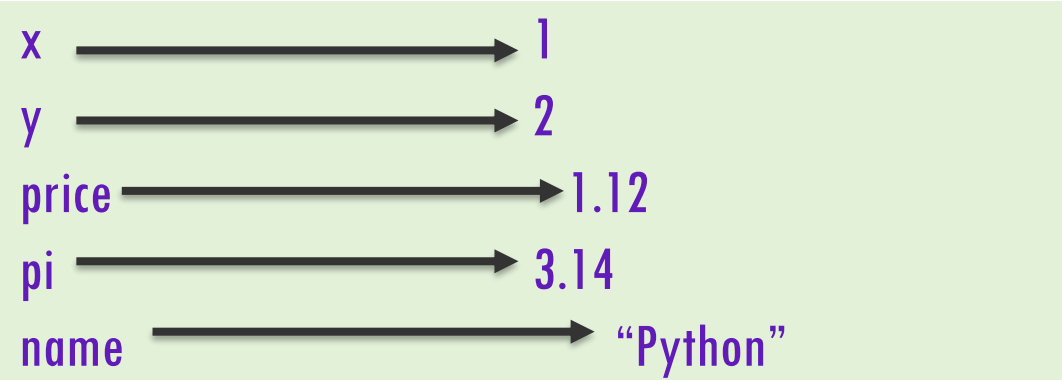
$$x = \frac{-b + \sqrt{b^2 - 4ac}}{2a}$$

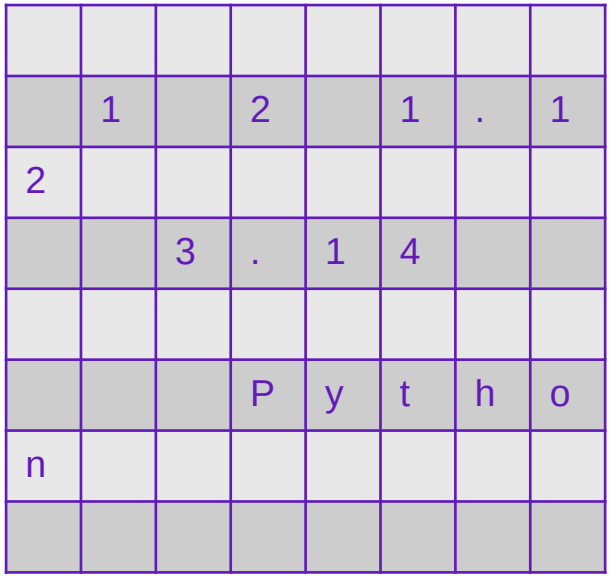
```
x = (-b + (b**2-4ac)**0.5)/2a
```

```
x = (-b + (b**2-4*a*c)**0.5)/(2*a)
```

Variable in memory

- **A variable is a name that refers to a value in the memory**
- Every variable has a **type**, a **size**, a **value** and a **location** in the computer's memory
  - size: the size of memory to store the value
- A state diagram (状态图) can be used for representing variable's state including name, type, size and value
- reassigned (重新赋值): 改变变量的指向





Simultaneous assignment statement

- A simultaneous assignment (同步赋值) statement allows us to calculate several values all at the same time:

  <var>, <var>, ..., <var> = <expr>, <expr>, ..., <expr>

  - It tells Python to evaluate all the expressions on the right-hand side;
    and then assign these values to the corresponding variables named on the left-hand side.
- For example,

  total, diff = x + y, x – y

  x+y and x-y will be computed and then assigned to sum and diff simultaneously. Finally total would get the sum of x and y and diff would get the difference.

```
49  x = 100
50  y = 49
51
52  total, diff = x+y, x-y
53  print(total)
54  print(diff)
```

```
149
51
```

Given x = 1, y =2, how to swap the values of x and y; i.e., x = 2, y = 1?

x, y = y, x

How about the following solution?

x= y

y =x

How about?

z = x

x = y

y = z

编程规范

实际项目开发的过程可能涉及到很多人，需要很长时间共同协作完成，必须制定更高的标准（管理3个人和管理10000个人的区别）

- 变量名：有意义的单词，单词和单词之间用_连接，一般用小写字母
  - book_id，car_number
- 文件名：命名规则和变量名一致。严格禁止：my-book.py等文件名
  - book.py
- 严格禁止使用Python语言自用单词用作变量名和文件名
  - print = 1, str = “hello”
  - print.py, id.py, str.py (导致系统自动带的print等函数被屏蔽)
- 空格：不要缩成一小撮
  - x+y-z/w 可以写的更清晰 x + y – z/w
  - 和英文写作中空格的标准一致
- VS Code格式自动化插件：Black Formatter
  - 安装后，右键Format document with

Zen of Python
Beautiful is better than ugly.
Explicit is better than implicit.

```
59  a,b,c=3,4,5
60  q=(a+b+c)/2
61  area=(q*(q-a)*(q-b)*(q-c))**0.5
62  print(area)
```

```
9   a, b, c = 3, 4, 5
10  q = (a + b + c) / 2
11  area = (q * (q - a) * (q - b) * (q - c)) ** 0.5
12  print(area)
```