Basic Syntax

講者：Isaac

Outline

- Python keyword
- Python identifier
- Python literal
- Basic syntax

Python keyword

Python keyword

- keywords is reserved word for python

```
and         del         from        not         while
as          elif        global      or          with
assert      else        if          pass        yield
break       except      import      print
class       exec        in          raise
continue    finally     is          return
def         for         lambda      try
```

Python keyword

- can use python to import keyword module to print keyword list

```
>>> import keyword
>>> keyword.kwlist
['False', 'None', 'True', 'and', 'as', 'assert', 'async',
'await', 'break', 'cla
ss', 'continue', 'def', 'del', 'elif', 'else', 'except',
'finally', 'for', 'from
', 'global', 'if', 'import', 'in', 'is', 'lambda', 'nonlocal',
'not', 'or', 'pas
s', 'raise', 'return', 'try', 'while', 'with', 'yield']
>>> keyword.iskeyword('if')
True
>>> keyword.iskeyword('IF')
False
>>>
```

Python identifier

Python identifier

- An identifier is a name given to entities like class, functions, variables
  - it helps to differentiate one entity from another
- rule of identifier
  - can be any alphabet(A-Z, a-z), digits(0-9), underscore(-)
  - cannot start with a digit
  - case-sensitive
  - cannot use keyword

Python identifier

name	valid or invalid
ab10c	valid
abc_DE	valid
_	valid
_abc	valid
99	invalid
x+y	invalid
for	invalid
a@	invalid
9abc	invalid

Python identifier

- can use .isidentifier() to detect if the identifier is valid or not

```
>>> print("abc".isidentifier()) # True
True
>>> print("99a".isidentifier()) # False
False
>>> print("_".isidentifier()) # True
True
>>> print("for".isidentifier()) # True - wrong output
```

Python literal

Python literal

- literal is a raw data given in a variable or constant
  - Numeric Literals
  - String literals
  - Bytes literals
  - Boolean literals
  - Special literals
  - ……

Numeric Literals

- Decimal
  - **30, 50**
- Binary
  - **0b11001, 0B111101**
- Octal
  - **0o156, 0o132**
- Hexadecimal
  - **0x12, 0x54**

String literals

- short string: use a pair of single quote to mark a string
- long string: use a pair of triple quote to mark a long string
- escape characters: a character which invokes an alternative interpretation on subsequent characters
- "+" character: concatenate two string
- "*" character: repeat string

String literals

Command Prompt - python

```
C:\Users\isaac>python
Python 3.6.8 (tags/v3.6.8:3c6b436a57, Dec 24 2018, 00:16:47) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> short_string = '123321'
>>> long_string = """123333333333333333333333"""
>>> new_line_string = '123 \\newline'
>>> concate_string = '123' + '456'
>>> print(short_string)
123321
>>> print(long_string)
123333333333333333333333
>>> print(new_line_string)
123 \newline
>>> print(concate_string)
123456
>>>
```

String literals

- Escape character

Escape Sequence	Meaning
\\	Backslash (\)
\'	Single Quote (')
\"	Double Quote (")
\a	ASCII Bell (BEL)
\b	ASCII Backspace (BS)
\f	ASCII Formfeed (FF)
\n	ASCII Linefeed (LF)
\N	Character named *name* in the Unicode database
\r	ASCII Charriage Return (CR)
\t	ASCII Horizontal Tab (TAB)
\uxxxx	Character with 16-bit hex value (Unicode only)
\Uxxxxxxxx	Character with 32 bit hex value (Unicode only)
\v	ASCII Vertical Tab(VT)
\ooo	Charcter with octal value ooo
\xhh	Character with hex value hh

Boolean literals

- Boolean literals
  - “True” or “False”

Command Prompt - python

```
C:\Users\isaac>python
Python 3.6.8 (tags/v3.6.8:3c6b436a57, Dec 24 2018, 00:16:47) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> a = True
>>> b = False
>>> a and b
False
>>> a or b
True
>>>
```

Basic syntax

statement in python

- Each line is a python statement
  - No need to use ";" to end a statement

```
a = 1
b = 2

print(a+b)
```

```
3
```

statement in python

- Comments in python
  - use # to do a single line comment
  - use pair of ''' to do multiple line comments

```
# this is a single line comment
# this is a single line comment


'''
this is multiple line comments


'''
```

- Multiple line statement
  - use backslash(\) to do multiple line statement

```

if a < 10 and b > 5 \
     and c < 6 and d > -1:
         print('do something')
```

statement in python

- Python use indentation to represent a block of code
  - Same indentation mean these code are in the same block
  - Usually use "tab" or 4 space to do indentation
  - Do not mix 'tab' and 4 space in coding. This may cause indentation error

```
for i in range(10):
    if i > 5:
        continue
    else:
        for j in range(10):
            print(j)

```

print function

- Python use print function to show information on screen

```python
print('hello python!')

a = 10
print(a)

print('hello python!', end='')
print('abc')
```

```
hello python!
10
hello python!abc
```

input function

- Python use input function to make user input something from keyboard

```
1 my_name = input('please input your name')
2 print('your name is')
3 print(my_name)
```

```
please input your nameisaac
your name is
isaac
```