# 大学计算机入门课

**Class #3**

［教学目的］

- Know how to use the if statement.
- Know more string operations and methods.

［课程大纲］

- part1 – String Operations and Methods.
- Part2 – if statement
- part 3 – no if required vs. if statements
- part 4 – syntax of and, or, not

## part1 – String Operations and Methods.

- What we already know about Strings

| | |
|---|---|
| String addition | >>> “hello” + “ tmr”<br>‘hello tmr’ |

<table>
<tr><td>String scalar multiplication</td><td>>>> “hi” * 3<br>‘hihihi’</td></tr>
<tr><td>String index</td><td>>>> a = ‘hello’<br><table><tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td></tr><tr><td>‘h’</td><td>‘e’</td><td>‘l’</td><td>‘l’</td><td>‘o’</td></tr><tr><td>-5</td><td>-4</td><td>-3</td><td>-2</td><td>-1</td></tr></table></td></tr>
<tr><td>String length</td><td>>>> a = ‘hello’<br>>>> len(a)<br>5<br># the length of a string is the number of characters in this string.</td></tr>
<tr><td>Get index i from a string</td><td>>>> a = ‘hello’<br>>>> a[0]<br>‘h’</td></tr>
<tr><td>Get slice from the string</td><td>>>> a[i : j]<br>a[i] + a[I + 1] + … + a[j-1]<br>>>> a[ : j]<br>从开头到 index (j - 1)<br>>>> a[i : ]<br>从 index（i） 到最后</td></tr>
<tr><td>In</td><td>x in s → bool<br>produce True if and only if x is in s</td></tr>
</table>

- More string methods!

```
  convert an object into its string representation, if possible.
S.count(sub[, start[, end]]) -> int
  Return the number of non-overlapping occurrences of substring sub in
  string S[start:end].  Optional arguments start and end are interpreted
  as in slice notation.
S.find(sub[, i]) -> int
  Return the lowest index in S (starting at S[i], if i is given) where the
  string sub is found or -1 if sub does not occur in S.
S.index(sub) -> int
  Like find but raises an exception if sub does not occur in S.
S.isalnum() -> bool
  Return True if and only if all characters in S are alphanumeric
  and there is at least one character in S.
S.isalpha() -> bool
  Return True if and only if all characters in S are alphabetic
  and there is at least one character in S.
S.isdigit() -> bool
  Return True if and only if all characters in S are digits
  and there is at least one character in S.
S.islower() -> bool
  Return True if and only if all cased characters in S are lowercase
  and there is at least one cased character in S.
S.isupper() -> bool
  Return True if and only if all cased characters in S are uppercase
  and there is at least one cased character in S.
S.lower() -> str
  Return a copy of the string S converted to lowercase.
S.lstrip([chars]) -> str
  Return a copy of the string S with leading whitespace removed.
  If chars is given and not None, remove characters in chars instead.
S.replace(old, new) -> str
  Return a copy of string S with all occurrences of the string old replaced
  with the string new.
S.rstrip([chars]) -> str
  Return a copy of the string S with trailing whitespace removed.
  If chars is given and not None, remove characters in chars instead.
S.split([sep]) -> list of str
  Return a list of the words in S, using string sep as the separator and
  any whitespace string if sep is not specified.
S.strip([chars]) -> str
  Return a copy of S with leading and trailing whitespace removed.
  If chars is given and not None, remove characters in chars instead.
  S.swapcase() -> str
    Return a copy of S with uppercase characters converted to lowercase
    and vice versa.
  S.upper() -> str
    Return a copy of the string S converted to uppercase.
```

## Part 2 – if statement

- 现在我们要考虑一个 function 要对于不同的情况实行不同的规则。

```
def description(x):
    if x > 0:
        print("I'm a positive number")
    elif x == 0:
        print("I'm Zero")
    else:
        print("I'm a negative number")
```

run it：

```
>>> description(5)
    I'm a positive number
>>> description(0)
    I'm Zero
>>> description(-5)
    I'm a negative number
```

格式：

if ... (is True):
    doA()
elif ... (else if ... is True):
    doB()
else:
    doC()

（注：一个 code 可以没有 elif / else）

对比 if and elif：

```
def description_str(word):
    if is_start_with_digit(word):
        print("I'm starting with digit")
    if len(word) == 5:
        print("I have length of 5!")
```

```
>>> description_str("1hihi")
    I'm starting with digit
    I have length of 5!
```

```
def description_str(word):
    if is_start_with_digit(word):
        print("I'm starting with digit")
    elif len(word) == 5:
        print("I have length of 5!")
```

```
>>> description_str("1hihi")
I'm starting with digit
```

we can also visualize it! Try it!

总结：当我们的 code 实行完第一个 if 后见到 elif / else，将不会进入到 elif / else 中的命令。

见到 if，将会继续阅读其中的命令。

- 在一个 function 的 body 里，我们可以有很多个 if / elif，
  并且每一个 if statement 里我们还可以套有另外的一个 if statement。
  For example, let's consider the following docstring:

```
def convert_case(s):
    """(str) -> str

    Return s in upper/lower case alphabet if it maked up by lower/upper
    case alphabet, else return s itself.

    >>> convert_case("apple")
    'APPLE'
    >>> convert_case("apple1")
    'apple1'
    >>> convert_case("APP")
    'app'
    """
```

如果 s 是由字母组成的，那么它还有三种可能性

1. 由小写字母组成；
2. 由大写字母组成；
3. 混合组成；

所以它的 body 将会出现一个 if statement 套有另外一个 if statement 的情况

```
    if s.isalph():
        if s.isupper():
            return s.lower()
        elif s.islower():
            return s.upper()
        else:
            return s
    else:
        return s
```

Now, let's do more about describe string.
Consider it case by case!!!

```
def describe_str(s):
    if len(s) == 0:
        print("This is an empty string")
    elif s.isdigit():
        print("This string formed by digits")
    elif s.isalpha():
        if s.isupper():
            print("This string formed by upper alpha")
        elif s.islower():
            print("This string formed by lower alpha")
        else:
            print("This string formed by upper and lower alpha")
    elif s.isalnum():
        print("This string formed by digits and alpha")
    else:
        print("This string contains special character(s).")
```

don't remember to test the code!

Try another function：
Converting numerical grade to letter grade:

| Percentage | Letter Grade |
|---|---|
| 90-100 | A+ |
| 85-89 | A |
| 80-84 | A- |
| 77-79 | B+ |
| 73-76 | B |
| 70-72 | B- |
| 67-69 | C+ |
| 63-66 | C |
| 60-62 | C- |
| 57-59 | D+ |
| 53-56 | D |
| 50-52 | D- |
| 0-49 | F |

- 简化 code
  当一个 function 中含有很多 else 时，有时我们可以省略掉 else。
  想想看 examples：

```
    if s.isalph():
        if s.isupper():
            return s.lower()
        elif s.islower():
            return s.upper()
        else:
            return s
    else:
        return s
```

可以简化成

```
136     if s.isalph():
137         if s.isupper():
138             return s.lower()
139         elif s.islower():
140             return s.upper()
141     return s
```

这是因为如果我们的 code 运行进行到：

1. 137 + 138 行，说明 s 由大写字母组成，这时我们 return s.lower()。此时，运行终止。
2. 139 + 140 行，说明 s 由小写字母组成，这时我们 return s.upper()。此时，运行终止。
3. 141 行，说明 s 不是由大写 / 小写字母组成，这时我们 return s itself。

但是在 describe_str 里，我们可以省略吗？

```
def describe_str(s):
    if len(s) == 0:
        print("This is an empty string")
    elif s.isdigit():
        print("This string formed by digits")
    elif s.isalpha():
        if s.isupper():
            print("This string formed by upper alpha")
        elif s.islower():
            print("This string formed by lower alpha")
        else:
            print("This string formed by upper and lower alpha")
    elif s.isalnum():
        print("This string formed by digits and alpha")
    else:
        print("This string contains special character(s).")
```

如果我们省略最后一个 else，将会发生什么？
Let's see the output in the python shell.

```
>>> describe_str("123")
    This string formed by digits
    This string contains special character(s).
>>> describe_str("sjgj23")
    This string formed by digits and alpha
    This string contains special character(s).
```

这是因为 print 不会终止运行， 所以

>>> print("This string contains special character(s).")

always run!

在这种时候，我们并不能省略 else。

总结：如果在 else 一以上任意一个 case 中没有出现 return（或者终止运行的信号）时，

我们都不能省略这个 else。

## part 3 – no if required vs. if statement

most of the time, Boolean function can be written with if statement or without if statement.

- Consider the can_drink function in the previous lectures.

```
def can_drink(age):
    """(int) -> bool

    precondition: age >= 0

    Return True if age is 19 or more, and False otherwise.

    >>> can_drink(21)
    True
    >>> can_drink(16)
    False
    """

    return age >= 18
```

we can also write this function with if statement.

```
def can_drink(age):
    if age >= 18:
        return True
    else:
        return False
```

- Now, let's consider another function is_teenager, which return True iff age represents a teenager between 13 and 18 inclusive.
  - firstly, let's written down the documentation strings.

```
def is_teenager(age):
    """(int) -> bool
    Return True iff age represents a teenager between 13 and 18 inclusive.
    >>> is_teenager(4)
    False
    >>> is_teenager(16)
    True
    >>> is_teenager(19)
    False
    """
```

  - write the body without if statement

```
def is_teenager(age):
    """(int) -> bool
    Return True iff age represents a teenager between 13 and 18 inclusive.
    >>> is_teenager(4)
    False
    >>> is_teenager(16)
    True
    >>> is_teenager(19)
    False
    """

    return 13 <= age <= 18
```

  - write the body with if statement

```
def is_teenager(age):
    """(int) -> bool
    Return True iff age represents a teenager between 13 and 18 inclusive.
    >>> is_teenager(4)
    False
    >>> is_teenager(16)
    True
    >>> is_teenager(19)
    False
    """
    if age < 13:
        return False
    elif age > 18:
        return False
    else:
        return True
```

## part 4 – syntax of and, or, not

| | |
|---|---|
| If A() and B():<br>C() | C() 会执行仅当 A，B both True |
| If A() or B():<br>C() | C() 会执行如果 A 是 True 或者 B 是 True |
| If not A():<br>C() | C() 会执行仅当 A 是 False |

Some examples：

```
def len5_digits(s):
    """(str) -> bool

    Return True if s is made up of digits and has length of 5.
    """
    if s.isdigit() and len(s) == 5:
        return True
    else:
        return False
```

```
def len5_or_digits(s):
    """(str) -> bool

    Return True if s is made up of digits or has length of 5.
    """
    if s.isdigit() or len(s) == 5:
        return True
    else:
        return False
```

now, let's do some brain logics:

1. `True and True` T

2. `False and True` F
3. `1 == 1 and 2 == 1` F
4. `"test" == "test"` T
5. `1 == 1 or 2 != 1` T
6. `True and 1 == 1` T
7. `False and 0 != 0` F
8. `True or 1 == 1` T
9. `"test" == "testing"` F
10. `1 != 0 and 2 == 1` F
11. `not (True and False)` T
12. `not (1 == 1 and 0 != 1)` F
13. `not (10 == 1 or 1000 == 1000)` F
14. `not (1 != 10 or 3 == 4)` F
15. `not ("testing" == "testing" and "Zed" == "Cool Guy")` T
16. `1 == 1 and (not ("testing" == 1 or 1 == 0))` T
17. `"chunky" == "bacon" and (not (3 == 4 or 3 == 3))` F
18. `3 == 3 and (not ("testing" == "testing" or "Python" == "Fun"))` F
19. `not("123qwe".isalpha() or not(1 == 2 or "123".isdigit())) and True` T

process:

1. find an equality test (== or !=) and replace it with True or False
2. fin each and/or inside parentheses and solve them and replace the whole part by its truth.
3. Find each not and invert it.
4. Find any remaining and/or and solve it.
5. Get your final result!