# List ??

??????????????????????????????????????????(?)?????

```
a = [1, 2, 3, 4, 5]
b = ['mango', 'pineapple', 'orange']
```

? Python ??List ? String ??????????????????????????

1. ???? `for` ??????
2. ???? indexing
3. ?? `len()` ?????????
4. ??????? `+` ???
5. ?? `in` ???????????????

List ? String ?????String ??????? (immutable)?List ?????? (mutable)?

## List methods

- `list[index] = x` ???? index ???

### list.append()

```
numbers = [1, 2, 3, 4]
numbers.append(5)
print(numbers)


# output: [1, 2, 3, 4, 5]
```

### list.insert()

```
animals = ["cat", "dog", "fish"]
animals.insert(1, "monkey")
print(animals)

# output: ["cat", "monkey", "dog", "fish"]

animals = ["cat", "dog", "fish"]
```

```
animals.insert(200, "monkey")
print(animals)

# output: ["cat", "dog", "fish", "monkey"]
```

# list.extend()

???? Lists

```
things = ["John", 42, True]
other_things = [0.0, False]
things.append(other_things)
print(things)

# output: ["John", 42, True, [0.0, False]]

things = ["John", 42, True]
other_things = [0.0, False]
things.extend(other_things)
print(things)

# output: ["John", 42, True, 0.0, False]
```

```
# This function accepts two variables, each containing a list of years.
# A current "recent_first" list contains [2022, 2018, 2011, 2006].
# An older "recent_last" list contains [1989, 1992, 1997, 2001].
# The lists need to be combined with the years in chronological order.
def record_profit_years(recent_first, recent_last):

    # Reverse the order of the "recent_first" list so that it is in
    # chronological order.
    recent_first.reverse()

    # Extend the "recent_last" list by appending the newly reversed
    # "recent_first" list.
    recent_last.extend(recent_first)

    # Return the "recent_last", which now contains the two lists
    # combined in chronological order.
    return recent_last
```

```
# Assign the two lists to the two variables to be passed to the
# record_profit_years() function.
recent_first = [2022, 2018, 2011, 2006]
recent_last = [1989, 1992, 1997, 2001]




# Call the record_profit_years() function and pass the two lists as
# parameters.
print(record_profit_years(recent_first, recent_last))
# Should print [1989, 1992, 1997, 2001, 2006, 2011, 2018, 2022]
```

## list.remove()

> Note: If there are two of the same element in a list, the ***.remove()*** method only removes the **first** instance of that element and not all occurrences.

```
booleans = [True, False, True, True, False]

booleans.remove(False)   # Removes the first False value
print(booleans)

# output: [True, True, True, False]

booleans.remove(False)   # Removes the other False value
print(booleans)

# output: [True, True, True]

booleans.remove(False)   # ValueError! No more False values to remove
```

## list.pop()

```
fruits = ["apple", "orange", "banana", "peach"]
last_fruit = fruits.pop()  # takes the last element
print(last_fruit)

# output: "peach"
```

```
second_fruit = fruits.pop(1)  # takes the second element ( = index 1)
print(second_fruit)

# output: "orange"

print(fruits)  # only fruits that have not been "popped"
               # are still in the list

# output: ["apple", "banana"]
```

## list.clear()

```
decimals = [0.1, 0.2, 0.3, 0.4, 0.5]
decimals.clear()  # remove all values!
print(decimals)

# output: []
```

## list.count()

```
grades = [7.8, 10.0, 7.9, 9.5, 10.0, 6.5, 9.8, 10.0]
n = grades.count(10.0)
print(n)

# output: 3
```

## list.index()

> “ Note: it only returns the index of the first occurrence of a list item.

```
friends = ["John", "James", "Jessica", "Jack"]
position = friends.index("Jessica")
print(position)

# output: 2
```

## list.sort() and list.reverse()

```
values = [10, 4, -2, 1, 5]

values.reverse()
print(values)  # list is reversed

# output: [5, 1, -2, 4, 10]

values.sort()
print(values)  # list is sorted

# output: [-2, 1, 4, 5, 10]
```

```
values = [10, 4, -2, 1, 5]

values.sort(reverse=True)
print(values)  # list is sorted in reverse order

# output: [10, 5, 4, 1, -2]
```

## list.copy()

```
values_01 = [1, 2, 3, 4]
values_02 = values_01  # not an actual copy: same list object!

values_02.append(5)  # we modify the "values_02" list...
print(values_01)     # ... but changes appear also in "values_01"
                     #     because they reference the same list!

# output: [1, 2, 3, 4, 5]



values_01 = [1, 2, 3, 4]
values_02 = values_01.copy()  # create an independent copy!

values_02.append(5)  # we modify the "values_02" list...
print(values_01)     # ... and changes DO NOT appear in "values_01"
                     #     because it is a copy!

# output: [1, 2, 3, 4]
```

# List functions

- `sorted()` ?????????????? data type?????????????? Key ?????
- `min()` ??????
- `max()` ??????
- `map(function, iterable)` Python - map() function
- `zip(*iterables)` ????? List ????? Tuple ????

## sorted()/min()/max()

```
time_list = [12, 2, 32, 19, 57, 22, 14]
print(sorted(time_list))
print(time_list)

names = ["Carlos", "Ray", "Alex", "Kelly"]
print(sorted(names))  # Output ['Alex', 'Carlos', 'Kelly', 'Ray']
print(names)          # Output ['Carlos', 'Ray', 'Alex', 'Kelly']
print(sorted(names, key=len)) # Output ['Ray', 'Alex', 'Kelly', 'Carlos']

time_list = [12, 2, 32, 19, 57, 22, 14]
print(min(time_list))
print(max(time_list))
```

## map()

Use `map()` and convert the map object to a list so we can print all the results at once.

```
# A simple function to add 1 to a given number
def add_one(number):
    return number + 1

# A list of numbers
numbers = [1, 2, 3, 4, 5]

# Use map to apply the function to each element in the list
result = map(add_one, numbers)

# Convert the map object to a list to print the result
print(list(result))
```

```
# Outputs: [2, 3, 4, 5, 6]
```

# zip()

Use `zip()` to combine a list of names and ages into a list of tuples, and print all the tuples at once.

```
# 基本 zip() 教學範例
>>> x = ['a', 'b', 'c']
>>> y = [1,   2,   3]
>>> zipped = zip(x, y)
>>> type(zipped) # 回傳的是一個 'zip' 物件，它是可迭代的
<class 'zip'>
>>> zipped
<zip object at 0x108e8bc80>

## 用 loop 遍歷 zip 物件內容
>>> for i in zip(x, y):
...     print(i)
('a', 1)
('b', 2)
('c', 3)

# 也可用 list() 或 set() 將迭代器轉換成其他資料型態
>>> list(zip(x, y))
[('a', 1), ('b', 2), ('c', 3)]
>>> set(zip(x, y))
{('c', 3), ('b', 2), ('a', 1)}
```

```
# Two lists
names = ["Alice", "Bob", "Charlie"]
ages = [25, 30, 35]

# Use zip to combine the lists
combined = zip(names, ages)

# Convert the zip object to a list to print the result
print(list(combined))

# Outputs: [('Alice', 25), ('Bob', 30), ('Charlie', 35)]
```

# Extracting from a list

```
# A element from a list
username_list = ["elarson", "fgarcia", "tshah", "sgilmore"]
print(username_list[2])

# one-liner
print(["elarson", "fgarcia", "tshah", "sgilmore"][2])

# A slice from a list
username_list = ["elarson", "fgarcia", "tshah", "sgilmore"]
print(username_list[0:2])
```

# List with Loop

```
animals = ["Lion", "Zebra", "Dolphin", "Monkey"]
chars = 0
for animal in animals:
  chars += len(animal)

print("Total characters: {}, Average length: {}".format(chars, chars/len(animals)))

# Output: Total characters: 22, Average length: 5.5
```

`enumerate()` ???????????????? tuple(??)??????????????????????????????????

```
winners = ["Ashley", "Dylan", "Reese"]
for index, person in enumerate(winners):
  print("{} - {}".format(index + 1, person))

# Output:
#1 - Ashley
#2 - Dylan
#3 - Reese
```

Output by line + 2 "\n"

```
IDs = ["001","002","003","004"]
print("\n\n".join([id for id in IDs]))
```

### For + If

```
mylist = [1, 4, 7, 8, 20]

newlist = [x for x in mylist if x % 2 == 0]
print(newlist)
```

### Range()

```
mylist = ["a", "b", "c", "d", "e", "f", "g"]

for x in range(2, len(mylist) - 1):
    print(mylist[x])
```

# List comprehensions

????????? list comprehension ???????????????? expression???????? for ??????????? for ? if ??????????? list???????? for ? if ????????????????

### for loop vs. list comprehensions

```
# For Loop
multiples = []
for x in range(1,11):
  multiples.append(x*7)

print(multiples)

# List comprehensions
multiples = [x*7 for x in range(1,11)]
print(multiples)
# Output [7, 14, 21, 28, 35, 42, 49, 56, 63, 70]
```

### Examples: Basic

```
languages = ["Python", "Perl", "Ruby", "Go", "Java", "C"]
lengths = [len(language) for language in languages]
```

```
print(lengths)

# Output [6, 4, 4, 2, 4, 1]
```

```
z = [x for x in range(0,101) if x % 3 == 0]
print(z)

# Output [0, 3, 6, 9, 12, 15, 18, 21, 24, 27, 30, 33, 36, 39, 42, 45, 48, 51, 54, 57, 60, 63, 66, 69, 72, 75, 78, 81, 84, 87, 90, 93, 96, 99]
```

## NOTE: ?????????List ??????

```
years = ["January 2023", "May 2025", "April 2023", "August 2024", "September 2025", "December 2023"]

updated_years = [year.replace("2023","2024") if year[-4:] == "2023" else year for year in years]

print(updated_years)
# Should print ["January 2024", "May 2025", "April 2024", "August 2024", "September 2025", "December 2024"]
```

```
years = ["January 2023", "May 2025", "April 2023", "August 2024", "September 2025", "December 2023"]

updated_years = [year.replace("2023","2024") for year in years if year[-4:] == "2023"]

print(updated_years)
# Should print ['January 2024', 'April 2024', 'December 2024']
```

## Examples: ???? Tuple ? List

```
# Create a list of tuples where each tuple contains the numbers 1, 2, and 3.
numbers = [(1, 2, 3) for _ in range(5)]

# numbers: [(1, 2, 3), (1, 2, 3), (1, 2, 3), (1, 2, 3), (1, 2, 3)]
```

## Examples: ???? List

```
def squares(start, end):
    return [ n * n for n in range(start, end+1) ]

print(squares(2, 3))    # Should print [4, 9]
print(squares(1, 5))    # Should print [1, 4, 9, 16, 25]
```

```
print(squares(0, 10))   # Should print [0, 1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

## Examples: ??????

```
def odd_numbers(x, y):
    return [n for n in range(x, y) if n % 2 != 0]


# Call the odd_numbers() function with two parameters.
print(odd_numbers(5, 15))
# Should print [5, 7, 9, 11, 13]
```

---

Revision #67
Created 20 June 2023 15:46:52 by Admin
Updated 11 December 2024 15:38:46 by Admin