04-if语句、Python字典、用户输入和while循环

if语句

In [52]:
```
cars = ['audi', 'bmw', 'subaru', 'toyota']
for car in cars:
    if car == 'bmw':
        print(car.upper())
    else:
        print(car.title())
```

```
Audi
BMW
Subaru
Toyota
```

条件测试

if语句的核心是一个值为 True 或 False 的表达式

- == 和 is 操作符的区别
- 检测是否不相等
- 大于、小于
- 使用and检查多个条件
- 使用or检查多个条件
- 检查是否在列表中： in 操作符
- 检查某个数是不是为0，不为0表示True，0表示False

In [3]:
```
car = 'bmw'
car == 'bmw'
```

```
True
True
```

== 操作符比较两个变量的值是否相等

In [53]:
```
print(cars == cars[:])
print([1, 2, 3] == list(range(1, 4)))
print((1, 2, 3) == tuple(range(1, 4)))
print(1 == 1.0)
print(1 == True)     # True == 1
print(0 == False)    # False == 0
```

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

is 比较两个变量是否指向同一个对象，对这两个变量使用 id() 函数应返回相同的结果

In [21]:
```
my_cars = cars
print(my_cars is cars)

x = 1
y = 1.0
print(x is y)
```

```
True
False
```

检测是否不相等： !=

In [22]:
```
requested_toppings = 'mushrooms'
if requested_toppings != 'anchovies':
    print("Hold the anchovies!")
```

```
Hold the anchovies!
```

检测不是同一个对象： is not

In [25]:
```
cars2 = cars[:]
print(cars2 is not cars)
```

```
True
```

数值比较大小： >，>=，<，<=

In [26]:
```
age = 18
print(age > 18)
print(age >= 18)
print(age < 18)
print(age <= 18)
```

```
False
True
False
True
```

使用 and 和 or 检查多个条件

In [28]:
```
age_0 = 22
age_1 = 18
print(age_0 >= 21 and age_1 >= 21)
print(age_0 >= 21 or age_1 >= 21)
```

```
False
True
```

使用 in 操作符检查特定值是否包含在列表中，not in 检查特定值是否不包含在列表中

In [34]:
```
requested_toppings = ['mushrooms', 'onions', 'pineapple']
print('mushrooms' in requested_toppings)
print('onions' not in requested_toppings)
```

```
True
False
```

简化多个条件

In [31]:
```
name = 'Jack'
pwd = '1234'
print((name, pwd) == ('Jack', '1234'))
```

```
True
```

In [33]:
```
x = 1
print(x == 0 or x == 1)
print(x in (0, 1))
```

```
True
True
```

if-else 语句

In [37]:
```
age = 17
if age >= 18:
    print("You are old enough to vote!")
    print("Have you registered to vote yet?")
else:
    print("Sorry, you are too young to vote.")
    print("Please register to vote as soon as you turn 18!")
```

```
Sorry, you are too young to vote.
Please register to vote as soon as you turn 18!
```

if...else 表达式

In [69]:
```
a, b = 10, 20
c = a if a else b
print(c)
```

```
10
```

if-elif-else 语句

In [38]:
```
age = 12
if age < 4:
    print("Your admission cost is $0.")
elif age < 18:
    print("Your admission cost is $5.")
else:
    print("Your admission cost is $10.")
```

```
Your admission cost is $5.
```

判断列表是否为空

In [54]:
```
requested_toppings = []

if requested_toppings:
    for topping in requested_toppings:
        print(f"Adding {topping}.")
    print("\nFinished making your pizza!")
else:
    print("Are you sure you want a plain pizza?")
```

```
Finished making your pizza!
```

判断字符串是否为空

In [59]:
```
msg = ''

if msg:
    print("msg is not empty")
else:
    print("msg is empty")
```

```
msg is empty
```

判断数值是否为0

In [60]:
```
x = 0.000

if x:
    print("x is not zero")
else:
    print("x is zero")
```

```
x is zero
```

In [65]:
```
a = 0 + 0.00j
print(a == 0)
if a:
    print("a is not zero")
else:
    print("a is zero")
```

```
True
a is zero
```

字典

In [1]:
```
alien_0 = {'color': 'green', 'points': 5}
print(alien_0['color'])
print(alien_0['points'])
```

```
green
5
```

添加键值对

In [2]:
```
alien_0['x_position'] = 0
alien_0['y_position'] = 25
print(alien_0)
```

```
{'color': 'green', 'points': 5, 'x_position': 0, 'y_position': 25}
```

创建空字典

In [4]:
```
alien_0 = {}
print(alien_0)
alien_0['color'] = 'green'
alien_0['points'] = 5
print(alien_0)
```

```
{}
{'color': 'green', 'points': 5}
```

修改字典中的值

In [5]:
```
alien_0['color'] = 'yellow'
print(f"The alien is now {alien_0['color']}.")
```

```
The alien is now yellow.
```

删除键值对

In [6]:
```
del alien_0['points']
print(alien_0)
```

```
{'color': 'yellow'}
```

由类似对象组成的字典

In [7]:
```
favorite_languages = {
    'jen': 'python',
    'sarah': 'c',
    'edward': 'ruby',
    'phil': 'python',
}

language = favorite_languages['sarah'].title()
print(f"Sarah's favorite language is {language}.")
```

```
Sarah's favorite language is C.
```

使用get()方法来返回默认值

In [8]:
```
alien_0 = {'color': 'green', 'speed': 'slow'}
print(alien_0['points'])
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
d:\workspaces\python_course\src\04-dicionaries-user-input-and-while-loops\04-dicionaries-user-input-and-while-loops.ipynb Cell 49 in <cell line: 2>()
      <a href='vscode-notebook-cell:/d%3A/workspaces/python_course/src/04-dicionaries-user-input-and-while-loops/04-dicionaries-user-input-and-while-loops.ipynb#Y102sZmlsZQ%3D%3D?line=0'>1</a> alien_0 = {'color': 'green', 'speed': 'slow'}
----> <a href='vscode-notebook-cell:/d%3A/workspaces/python_course/src/04-dicionaries-user-input-and-while-loops/04-dicionaries-user-input-and-while-loops.ipynb#Y102sZmlsZQ%3D%3D?line=1'>2</a> print(alien_0['points'])

KeyError: 'points'
```

In [10]:
```
print(alien_0.get('points', 'No point value assigned.'))
```

```
No point value assigned.
```

遍历字典键值对

In [11]:
```
user_0 = {
    'username': 'efermi',
    'first': 'enrico',
    'last': 'fermi',
}

for key, value in user_0.items():
    print(f'\nKey:{key}')
    print(f'Value:{value}')
```

```

Key:username
Value:efermi

Key:first
Value:enrico

Key:last
Value:fermi
```

In [12]:
```
for name, language in favorite_languages.items():
    print(f"{name.title()}'s favorite language is {language.title()}.")
```

```
Jen's favorite language is Python.
Sarah's favorite language is C.
Edward's favorite language is Ruby.
Phil's favorite language is Python.
```

遍历字典的键

In [13]:
```
for name in favorite_languages.keys():
    print(name.title())
```

```
Jen
Sarah
Edward
Phil
```

In [14]:
```
print(sorted(favorite_languages.keys()))
```

```
['edward', 'jen', 'phil', 'sarah']
```

遍历字典中的值

In [15]:
```
for language in favorite_languages.values():
    print(language.title())
```

```
Python
C
Ruby
Python
```

使用set()函数剔除重复项

In [16]:
```
print(set(favorite_languages.values()))
```

```
{'ruby', 'c', 'python'}
```

set集合数据结构

- 不包含重复的元素
- 和dict同样使用 { } 来定义，但不是键值对
- 空的集合用 set() 表示， {} 表示的是空的字典

嵌套的数据结构： 字典列表

In [17]:
```
alien_0 = {'color': 'green', 'points': 5}
alien_1 = {'color': 'yellow', 'points': 10}
alien_2 = {'color': 'red', 'points': 15}

aliens = [alien_0, alien_1, alien_2]
aliens
```

Out[17]:
```
[{'color': 'green', 'points': 5},
 {'color': 'yellow', 'points': 10},
 {'color': 'red', 'points': 15}]
```

在字典中存储列表

In [1]:
```
pizza = {
    'crust': 'thick',
    'toppings': ['mushrooms', 'extra cheese'],
}

print(pizza)
```

```
{'crust': 'thick', 'toppings': ['mushrooms', 'extra cheese']}
```

In [2]:
```
favorite_languages = {
    'jen': ['python', 'ruby'],
    'sarah': ['c'],
    'edward': ['ruby', 'go'],
    'phil': ['python', 'haskell'],
}

print(favorite_languages)
```

```
{'jen': ['python', 'ruby'], 'sarah': ['c'], 'edward': ['ruby', 'go'], 'phil': ['python', 'haskell']}
```

在字典中存储列表

In [3]:
```
users = {
    'aeinstein': {
        'first': 'albert',
        'last': 'einstein',
        'location': 'princeton',
        },
    'mcurie': {
        'first': 'marie',
        'last': 'curie',
        'location': 'paris',
        },
}

print(users)
```

```
{'aeinstein': {'first': 'albert', 'last': 'einstein', 'location': 'princeton'}, 'mcurie': {'first': 'marie', 'last': 'curie', 'location': 'paris'}}
```

用户输入

- 使用input函数，用户可以在控制台进行输入。

- input()接受一个参数，要向用户显示的提示（prompt）或说明。
- input接受用户的输入后，将返回包含用户输入内容的字符串
- 用户输入的数字也是 str 类型，进行数值相关操作时必须转换成对应的数值类型，例如使用 int() 函数或 float() 函数

In [4]:
```
message = input("Tell me something, and I will repeat it back to you: ")
```

In [5]:
```
print(message)
```

```
Hello
```

In [6]:
```
age = input("How old are you?")
```

In [7]:
```
age >= 18
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
d:\workspaces\python_course\src\04-dicionaries-user-input-and-while-loops\04-dicionaries-user-input-and-while-loops.ipynb Cell 74 in <cell line: 1>()
----> <a href='vscode-notebook-cell:/d%3A/workspaces/python_course/src/04-dicionaries-user-input-and-while-loops/04-dicionaries-user-input-and-while-loops.ipynb#Y140sZmlsZQ%3D%3D?line=0'>1</a> age >= 18

TypeError: '>=' not supported between instances of 'str' and 'int'
```

In [8]:
```
int(age) >= 18
```

Out[8]:
```
True
```

while循环

In [9]:
```
unconfirmed_users = ['alice', 'brian', 'candace']
confirmed_users = []

while unconfirmed_users:
    current_user = unconfirmed_users.pop()
    print(f"Verifying user: {current_user.title()}")
    confirmed_users.append(current_user)

print("\nThe following users have been confirmed:")
for confirmed_user in confirmed_users:
    print(confirmed_user.title())
```

```
Verifying user: Candace
Verifying user: Brian
Verifying user: Alice

The following users have been confirmed:
Candace
Brian
Alice
```

match/case 语句

- Python 3.10引入了新的 match/case 语句
- match/case 语句类似于C和Java中的 switch/case 语句，但更加强大，可以匹配更多的模式
- 具体可以参考官方文档PEP 636 – Structural Pattern Matching: Tutorial (https://peps.python.org/pep-0636/)
- 或者参考《Fluent Python》一书中这些小节：
  - Pattern Matching with Sequences
  - Pattern Matching with Mapping
  - Pattern Matching Class Instances
  - Pattern Matching in lis.py: A Case Study

In [15]:
```
# 输入格式为: action object
# 例如：
# move north
# get sword
# attack orc
command = input("What are you doing next? ")
```

In [16]:
```
match command.split():
    case ["quit"]:
        print("Goodbye!")
        print("quit_game()")
    case ["look"]:
        print("current_room.describe()")
    case ["get", obj]:
        print(f"Get a {obj}")
    case ["go", direction]:
        print(f"Go to the {direction}")
    case _:
        print("Unknown command.")
```

```
Go to the south
```

习题：从两个列表来创建字典

有两个列表，可能有不同的长度。第一个由 keys 组成，第二个由 values 组成。

写一个函数 createDict(keys, values) ，返回由 keys 和 values 创建的 dictionary。如果没有足够的值，其余的键应该有一个 None 值。如果没有足够的键，就忽略其余的值。

Examples:

```
keys = ['a', 'b', 'c', 'd']
values = [1, 2, 3]
createDict(keys, values) # {'a': 1, 'b': 2, 'c': 3, 'd': None}

keys = ['a', 'b', 'c']
values = [1, 2, 3, 4]
createDic(keys, values) # {'a': 1, 'b': 2, 'c': 3}
```

In [ ]:
```
def createDict(keys, values):

    return None
```

习题： 括号匹配

写一个函数，接收一串括号，并确定括号的顺序是否有效。如果字符串是有效的，它应该返回True，如果是无效的，它应该返回False。

例如：

```
"(){}[]" => True
"([{}])" => True
 "(}" => False
 "[(])" => False
"[({})](]" => False
```

In [ ]:
```
def match_brackets(s):
    stack = []
    brackets = {'(': ')', '[': ']', '{': '}'}
    for c in s:
        if c in brackets:
            stack.append(c)
        elif c in brackets.values():
            if not stack or brackets[stack.pop()] != c:
                return False
    return not stack
```