[Bài đọc] Cấu trúc điều kiện if-else

Giới thiệu

Có những tình huống xảy ra trong đời sống khi chúng ta cần phải đưa ra một số quyết định và dựa trên những quyết định này, chúng ta sẽ quyết định mình nên làm gì tiếp theo. Các tình huống tương tự cũng phát sinh trong lập trình, nơi chúng ta cần đưa ra một số quyết định và dựa trên những quyết định này, chúng ta sẽ thực thi khối mã tiếp theo.

Việc ra quyết định là cần thiết khi chúng ta muốn thực thi một đoạn code chỉ khi nó thỏa mãn kiện nào đó. Sau đây chúng ta sẽ đi tìm hiểu về các câu lệnh if trong Python, mỗi mục đều có ví dụ và diễn giải cụ thể để bạn hiểu rõ.

Nội dung bài đọc sẽ có các phần chính sau:

- Câu lệnh if
- Câu lệnh if…else
- Câu lệnh if lồng nhau
- Câu lệnh if…elif…else

Câu lệnh if

Câu lệnh if là câu lệnh ra quyết định đơn giản nhất. Nó được sử dụng để quyết định xem một câu lệnh hoặc khối câu lệnh nhất định sẽ được thực thi hay không, tức là nếu một điều kiện nhất định là đúng thì một khối câu lệnh được thực thi hoặc không.

Cú pháp:

```
if condition:

  # Statements to execute if condition is true
```

Điều kiện (condition) sau khi đánh giá sẽ là true hoặc false (boolean). Nếu giá trị là True thì nó sẽ thực thi khối lệnh bên dưới nó.

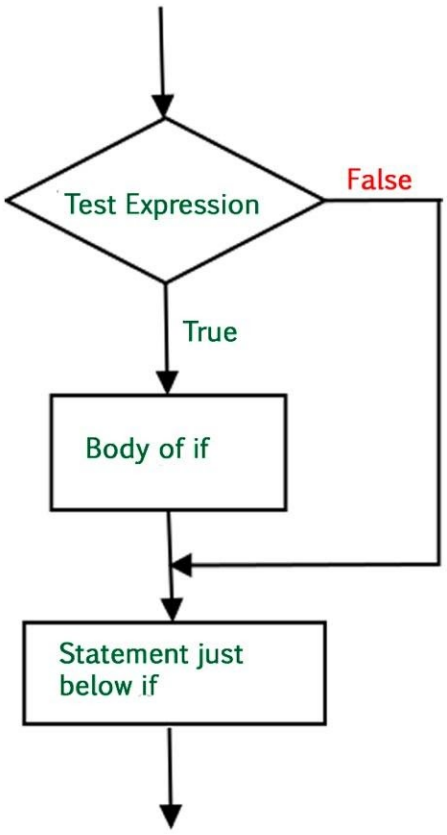
Python sử dụng thụt đầu dòng để xác định một khối. Vì vậy, khối bên dưới câu lệnh if sẽ được xác định như thể hiện trong ví dụ dưới đây:

```
if condition:

  statement1

statement2
```

Ở đây nếu điều kiện (condition) là True, khối if sẽ chỉ xem khối lệnh 1 (statement1) nằm bên trong khối của nó




```
i = 10
if (i > 15):
   print ("10 is less than 15")
print ("I am Not in if")
```

Kết quả:

```
I am Not in if
```

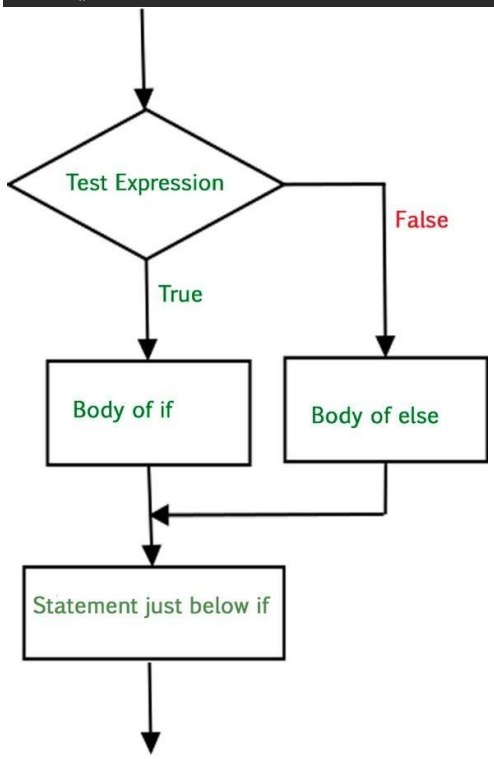
Điều kiện trong câu lệnh if là False nên khối lệnh bên dưới câu lệnh if không được thực thi

Câu lệnh if…else

Câu lệnh if cho chúng ta biết rằng nếu một điều kiện True, chương trình sẽ thực thi một khối câu lệnh và nếu điều kiện False thì không. Nhưng nếu chúng ta muốn làm điều gì đó khác nếu điều kiện False. Đây là câu lệnh khác. Chúng ta có thể sử dụng câu lệnh else với câu lệnh if để thực thi một khối mã khi điều kiện sai.

Cú pháp:

```
if (condition):
    # Executes this block if condition is true
else:
    # Executes this block if is false
```



Ví dụ:

```
i = 20;
if (i < 15):
    print ("i is smaller than 15")
    print ("i'm in if Block")
else:
    print ("i is greater than 15")
    print ("i'm in else Block")
print ("i'm not in if and not in else Block")
```

Kết quả:

```
i is greater than 15
i'm in else Block
i'm not in if and not in else Block
```

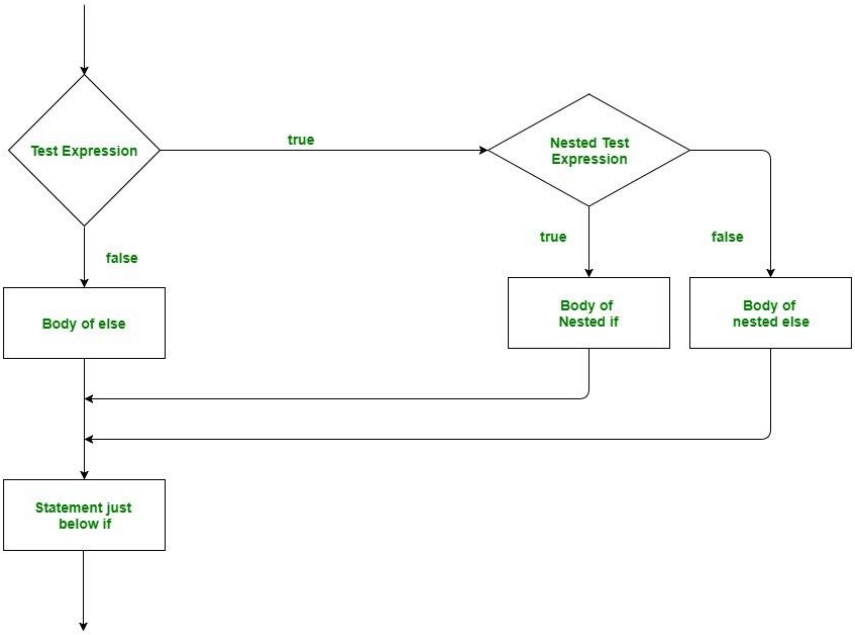
Khối mã theo sau câu lệnh else được thực thi vì điều kiện có trong câu lệnh if là False. Sau đó câu lệnh không nằm trong khối if…else sẽ được thực thi.

Câu lệnh if lồng nhau

Một if lồng nhau là một câu lệnh if là mục tiêu của một câu lệnh if khác. Các câu lệnh if lồng nhau có nghĩa là một câu lệnh if bên trong một câu lệnh if khác.

Cú pháp:

```
if (condition1):
   # Executes when condition1 is true
   if (condition2):
      # Executes when condition2 is true
   # if Block is end here
# if Block is end here
```



Ví dụ:

```
i = 10
if (i == 10):
    #  First if statement
    if (i < 15):
        print ("i is smaller than 15")
    # Nested - if statement
    # Will only be executed if statement above
    # it is true
    if (i < 12):
        print ("i is smaller than 12 too")
    else:
        print ("i is greater than 15")
```

Kết quả:

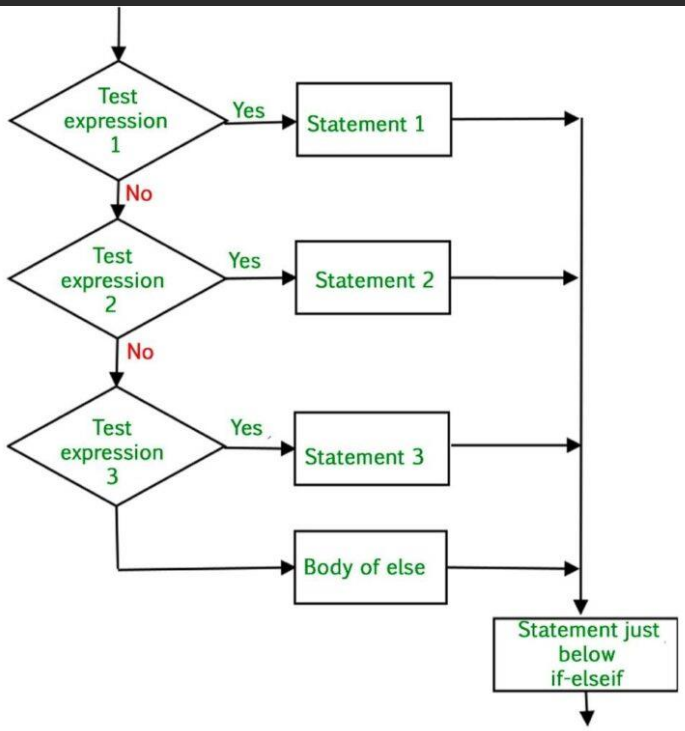
```
i is smaller than 15
i is smaller than 12 too
```

Câu lệnh if…elif…else

elif là viết gọn của else if, nó cho phép chúng ta kiểm tra nhiều điều kiện. Nếu điều kiện là sai, nó sẽ kiểm tra điều kiện của khối elif tiếp theo và cứ như vậy cho đến hết. Nếu tất cả các điều kiện đều sai nó sẽ thực thi khối lệnh của else. Chỉ một khối lệnh trong if…elif…else được thực hiện theo điều kiện. Có thể không có hoặc có nhiều elif, phần else là tùy chọn.

Cú pháp:

```
if (condition):
    statement
elif (condition):
    statement
.
.
else:
    statement
```



Ví dụ

```
i = 20
if (i == 10):
    print ("i is 10")
elif (i == 15):
    print ("i is 15")
```

```
elif (i == 20):
    print ("i is 20")
else:
    print ("i is not present")
```

Kết quả:

```
i is 20
```

[Bài đọc] Các lỗi thường gặp trong câu lệnh if-else

Lỗi 1: Sai cú pháp

Ví dụ:

```
num = int(input("Give me a number: "))
if num > 0:
print(num, "is positive")
```

Kết quả:

```
File "main.py", line 3
    print(num, "is positive")
    ^
IndentationError: expected an indented block
```

Kết luận:

Như chúng ta đã biết, python sử dụng thụt đầu dòng để xác định một khối. Vì vậy, hãy chắc chắn rằng khối bên dưới câu lệnh if đã được viết đúng cú pháp:

```
if condition:
   # Statements to execute if condition is true
```

Lỗi 2: Sai cú pháp toán tử quan hệ

Ví dụ:

```
num = int(input("Give me a number: "))
if num = 0:
    print("You entered 0")
```

Kết quả:

```
File "main.py", line 2
```

```
    if num = 0:
           ^
SyntaxError: invalid syntax
```

Kết luận:

Câu lệnh if được sử dụng để quyết định xem một câu lệnh hoặc khối câu lệnh nhất định sẽ được thực thi hay không, tức là nếu một điều kiện nhất định là đúng thì một khối câu lệnh được thực thi hoặc không. Vì vậy, hãy chắc chắn rằng điều kiện đã được viết đúng cú pháp.

Lỗi 3: Không viết đầy đủ các điều kiện cho câu lệnh if

Ví dụ:

```
num = int(input("Give me a number: "))
if num > 0:
    print(num, "is positive")
```

Kết quả:

```
Give me a number: 0
>
Không trả về kết quả đối với những số <= 0
```

Kết luận:

Hãy lường trước được những trường hợp có thể sẽ xảy ra và viết nó một cách chặt chẽ nhất để kết quả sẽ luôn được trả về.

```
num = int(input("Give me a number: "))
if num > 0:
    print(num, "is positive")
if num < 0:
    print(num, "is negative")
if num == 0:
    print(num, "is zero")
```

Lỗi 4: Viết quá nhiều câu lệnh if

Ví dụ:

```
gas = input("What type of gas do you want? ")
if gas == "regular":
       print("Regular Gas:", 3.89*17)
if gas == "plus":
       print("Plus Gas:", 4.09*17)
if gas == "premium":
```

```
        print("Premium Gas:", 4.19*17)
if gas == "diesel":
        print("Diesel Gas:", 4.59*17)
else:
        print("Invalid Gas Type")
```

Kết luận:

Python sẽ đánh giá tất cả các câu lệnh if để xác định xem chúng có đúng không.

Trong khi một điều kiện trong câu lệnh if elif else đúng, Python sẽ ngừng đánh giá các điều kiện khác. Do đó, if elif else nhanh hơn các lệnh if.

```
gas = input("What type of gas do you want? ")
if gas == "regular":
        print("Regular Gas:",3.89*17)
elif gas == "plus":
        print("Plus Gas:",4.09*17)
elif gas == "premium":
        print("Premium Gas:",4.19*17)
elif gas == "diesel":
        print("Diesel Gas:",4.59*17)
else:
        print("Invalid Gas Type")
```

Lỗi 5: Viết quá nhiều điều kiện lồng nhauß

Nếu có thể, hãy tận dụng các toán tử logic để có được những dòng code gọn gàng và hiệu quả nhất.

Thay vì:

```
dayssent = int(input("How many days in a row have you sent your friend a
message?"))
daysreceived = int(input("How many days in a row has your friend sent you
a message? "))
if dayssent >= 100:
        if daysreceived >= 100:
                print("you receive a hundred emoji")
else:
        print("not there yet")
```

Nên viết thành:

```
dayssent = int(input("How many days in a row have you sent your friend a
message? "))
daysreceived = int(input("How many days in a row has your friend sent you
a message? "))
if dayssent >= 100 and daysreceived >= 100:
        print("you receive a hundred emoji")
else:
```

```
        print("not there yet")
```

[Bài đọc] Toán tử ba ngôi

Toán tử điều kiện (ba ngôi) – Tenary Operator

Giới thiệu

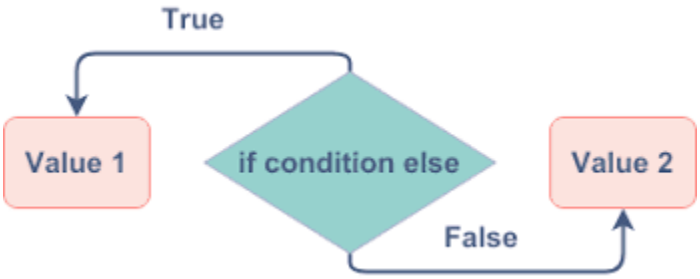
Toán tử ba ngôi là một loại biểu thức điều kiện trong Python, được sử dụng để chọn một trong hai giá trị dựa trên một điều kiện. Nó là bản thu nhỏ và ngắn gọn hơn câu lệnh if… else truyền thống.

Cú pháp:

```
statement_1 if expression else statement_2
```

`statement_1` được chọn nếu biểu thức kết quả là `True`. Nếu biểu thức kết quả là `False`, `statement_2` được chọn.

Bạn có thể cung cấp một giá trị, biến hoặc câu lệnh cho `statement_1 statement_2`



Ví dụ 1: Tìm số lớn trong hai số,

```
a, b = 2, 5
max = a if a > b else b
print(max)
```

Kết quả: 5

Toán tử ba ngôi trong ví dụ, chọn a hoặc b dựa trên điều kiện a>b đánh giá True hoặc False tương ứng. Vì a>b trả về False, nên lấy b là 5

Ví dụ 2: Viết các câu lệnh print dựa trên giá trị trả về của điều kiện, Python thực thi một trong các câu lệnh print.

```
a, b = 2, 5
```

```
print('Python') if a < b else print('Examples')
```

Kết quả: Python

Chạy chương trình, dựa trên điều kiện a <b trả về True nên câu lệnh đầu tiên được thực hiện.

Bạn có thể lồng một toán tử điều kiện ba ngôi trong một câu lệnh khác cũng sử dụng toán tử điều kiện ba ngôi.

Ví dụ 3: Trong ví dụ sau, chúng ta sẽ sử dụng toán tử điều kiện ba ngôi lồng nhau và tìm số lớn nhất.

```
a, b, c = 15, 100, 22

max = a if a > b and a>c else b if b>c else c

print(max)
```

Kết quả: 100

Chạy chương trình, sau mỗi keyword else, sẽ thực hiện toán tử ba ngôi để trả kết quả thỏa mãn với các điều kiện

Tổng kết

Qua bài viết này chúng ta đã tìm hiểu:

- Toán tử ba ngôi trong Python là gì?
- Cách sử dụng toán tử ba ngôi trong Python thông qua các ví dụ cơ bản
- Cách các câu lệnh thực thi bên trong toán tử

[Bài đọc] Coding Convention với biến, cấu trúc điều kiện, lặp

Giới thiệu

Coding Convention hiểu đơn giản là một tập các quy tắc chuẩn khi lập trình một ngôn ngữ cụ thể. Những quy tắc này bao gồm: cách tổ chức file, thụt lề, hướng dẫn comment, khai báo biến, quy tắc viết câu lệnh, khoảng trắng, quy tắc đặt tên, nguyên tắc lập trình phần mềm, …

Tiêu chuẩn viết code Python chính thức là PEP 8 (Python Enhancements Proposal 8)

PEP8 là gì ?

PEP-8 là một chuẩn code dùng trong ngôn ngữ lập trình python. Chúng được viết vào năm 2001 bởi Guido van Rossum, Barry Warsaw và Nick Coghlan. PEP 8 là tài liệu dành cho lập trình viên mới bắt đầu hoặc trình độ trung cấp mà đây không phải topic cho lập trình viên cao cấp

1. Quy tắc đặt tên

- Tên Class nên viết hoa mỗi chữ cái đầu tiên của từ.Cách viết này gọi là camel case

Vd: MyClass, SomeExampleClass

- Tên hàm và biến nên viết thường và tách từ bằng ký tự _

Vd: function, my_function, my_variable

- Tên hằng số viết chữ in hoa, tách từ bằng ký tự _

Vd: CONSTANT, MY_CONSTANT

- Tên module viết thường, ngắn gọn, dễ hiểu

Vd: support.py

2. Thụt lề : Sử dụng 4 khoảng trắng liên tiếp cho mỗi cấp lùi đầu dòng

3. Import:

- Mỗi dòng chỉ nên import 1 thư viện.
- Các lệnh import nên được đặt ở đầu file , trước khi bạn khai báo các module, các biến và hằng số
- Thứ tự các nhóm import như sau:

1. Import các thư viện chuẩn
2. Import các thư viện của nhà cung cấp thứ ba
3. Import các thư viện và ứng dụng cục bộ

Nên viết:

```
import  flask

import  app
```

Không nên viết:

```
import  flask,  app
```

- Tuy nhiên, nếu import 2 module con từ 1 thư viện, bạn có thể import trong 1 dòng

from Flask import flask, app

4. Chiều dài tối đa của 1 dòng code :

- Chiều dài tối đa của 1 dòng code là 79 kí tự
- Đối với khối văn bản dài(comment) : nên giới hạn chỉ 72 kí tự
- Sử dụng 1 câu comment hoàn chỉnh, bắt đầu bằng chữ cái viết hoa, thay đổi nội dung comment khi thay đổi đoạn code
- Đối với dòng code dài thì nên ngắt ra thành nhiều dòng ngắn : bằng cách đưa các biểu thức vào các cặp ngoặc đơn, cắt dòng khi hoàn thành biểu thức và thêm dấu "\" rồi xuống dòng viết tiếp

Nên viết:

```
with open('/path/to/some/file/you/want/to/read')  as  file_1,  \

                 open('/path/to/some/file/being/written' ,   'w')  as
file_2:

                 file_2.write(file_1.read() )
```

Không nên viết:

```
with open('/path/to/some/file/you/want/to/read')  as  file_1,  \
       open('/path/to/some/file/being/written' ,   'w')  as  file_2:
       file_2.write(file_1.read() )
```

Nên viết:

```
income =  (gross_wages
             + taxable_interest
             +  (dividends   -   qualified_dividends)
              -  ira_decuction
              -  student_loan_interest)
```

Không nên viết:

```
income =  (gross_wages +
              taxable_interest +
             (dividends   -   qualified_dividends)  -
              ira_decuction  -
              student_loan_interest)
```

5. Tab hay Space

Trong Python thì Space (dấu cách) được ưu tiên hơn. Còn dấu Tab nên sử dụng trong các trường hợp mà code đã sử dụng sẵn Tab rồi.

*Lưu ý: Python 3 không cho phép sử dụng Tab và Space lẫn lộn trong cùng 1 file

6. Sử dụng dấu cách

Bạn nên tránh các dấu thừa trong các trường hợp sau:

- Ngay trong ngoặc

Ví dụ:

Sai `foo= (bar, buz  )`

Đúng `foo= (bar, buz)`

- Ở sau dấu phẩy hoặc trước ngoặc đóng

Ví dụ:

Sai `foo= (bar,  )`

Đúng `foo= (bar,)`

- Ngay trước dấu chấm phẩy , dấu phẩy hoặc hai chấm

Ví dụ:

Sai `if a is True  :  print(1  ,2);  x  ,  y  = 1  ,  2`

Đúng `if a is True:  print(1 , 2);  x ,  y  = 1 ,  2`

- Mỗi lần gán biến, chỉ cần một dấu cách

Ví dụ:

Sai
```
foo                                                 = 1
bar                                                  = 2
            this_is_a_really_long_name = 3
```

Đúng
```
foo = 1
bar = 2
this_is_a_really_long_name = 3
```

7. Sử dụng dấu nháy đơn / đôi cho chuỗi

Trong Python, việc để nội dung chuỗi trong nháy đơn hoặc nháy đôi là như nhau

Tuy nhiên khi viết comment code thì nên sử dụng nháy đôi nội dung

[Bài tập] Luyện tập với toán tử ba ngôi

Mục tiêu

Luyện tập sử dụng toán tử điều kiện ba ngôi (Tenary Operator)

Mô tả

Kiểm tra số nhập vào là số chẵn hoặc lẻ, sử dụng toán tử điều kiện ba ngôi

Hướng dẫn

Nhập vào 1 số nguyên bất kì. Kiểm tra số nguyên nhập vào có chia hết cho 2 hay không?

- Nếu chia hết cho 2: số chẵn
- Nếu không chia hết cho 2 có dư : số lẻ

Demo:

```
num_check = ?

# USING TERNARY OPERATOR
msg = "Even" if num_check %2 == 0 else "Odd"
print(msg)

# USING USUAL IF-ELSE
msg = ""
if(num_check%2  == 0):
msg = "Even"
else:
msg = "Odd"
print(msg)ß
```

[Bài tập] Kiểm tra số chẵn/lẻ

Mục tiêu

Luyện tập sử dụng khối lệnh if…elif…else trong lập trình.

Mô tả

Viết chương trình có khả năng kiểm tra tính chất chẵn lẻ của một số nhập vào. In ra màn hình số nhập vào là số chẵn, số lẽ hay không phải là các trường hợp trên.

Hướng dẫn

Sử dụng function: input() để nhập một số bất kì sau đó chuyển kiểu về float. Kiểm tra số nhập vào có chia hết cho 2 hay không?

- Nếu chia hết cho 2: số chẵn
- Nếu không chia hết cho 2 dư 1: số lẽ
- Ngoài ra: không phải số tự nhiên.

Demo:

```
a = input()
a = float(a)
if a % 2 == 0:
    print('số chẵn')
elif a % 2 == 1:
    print('số lẻ')
else:
    print('không phải số tự nhiên')
```

BÀI TẬP:

Bài 1:

Tình tiền bằng số km đã nhập vào từ bàn phím theo yêu cầu sau:

1km sẽ trả 5000d

Trong khoảng 1km sẽ tới <=5km trả (km-1)*4500+5000

Từ 5km tới <= 120km sẽ trả (km-5)*3500+4500*4+5000

Trên 120km sẽ trả (((km-5)*3500+4500*4+5000)*1/10)

Bài 2:

Tính tiền giờ sử dụng máy tính với giờ bắt đầu và kết thức như sau: (1h = 5000d)

Thời gian bắt đầu từ >= 0 và <= 7h, Nếu số h dùng => 7 thì số tiền sẽ là Số giờ dùng nhân * số tiền 1h * 300*60 sau đó chia cho 0.15,

nếu số h dùng < 7 thì số tiền là Số giờ dùng nhân * số tiền 1h * 300

Thời gian bắt đầu từ > 7 và <= 17h, Nếu số h dùng => 6 thì số tiền sẽ là Số giờ dùng nhân * số tiền 1h * 400*60 sau đó chia cho 0.1,

nếu số h dùng < 6 thì số tiền là Số giờ dùng nhân * số tiền 1h * 400

Thời gian bắt đầu từ > 17 và <= 24h, Nếu số h dùng => 4 thì số tiền sẽ là Số giờ dùng nhân * số tiền 1h * 350*60 sau đó chia cho 0.12,

nếu số h dùng < 4 thì số tiền là Số giờ dùng nhân * số tiền 1h * 350