THE HONG KONG POLYTECHNIC UNIVERSITY
香港理工大學

Department of Computing
電子計算學系

Fall, 2020

Python Programming Workshop

Workshop 2
Control Flow and Lists

Kevin Kam Fung Yuen
Department of Computing
Hong Kong Polytechnic University
kevin.yuen@polyu.edu.hk

Class Schedule (Tentative)

19 Sep: Python Programming Environment and Basics

26 Sep: Control Flow and Lists

3 Oct: Tuples and Dictionaries

10 Oct: Functions, Modules, OO and Packages

17 Oct: Scientific Computing and Plotting

Lecture notes

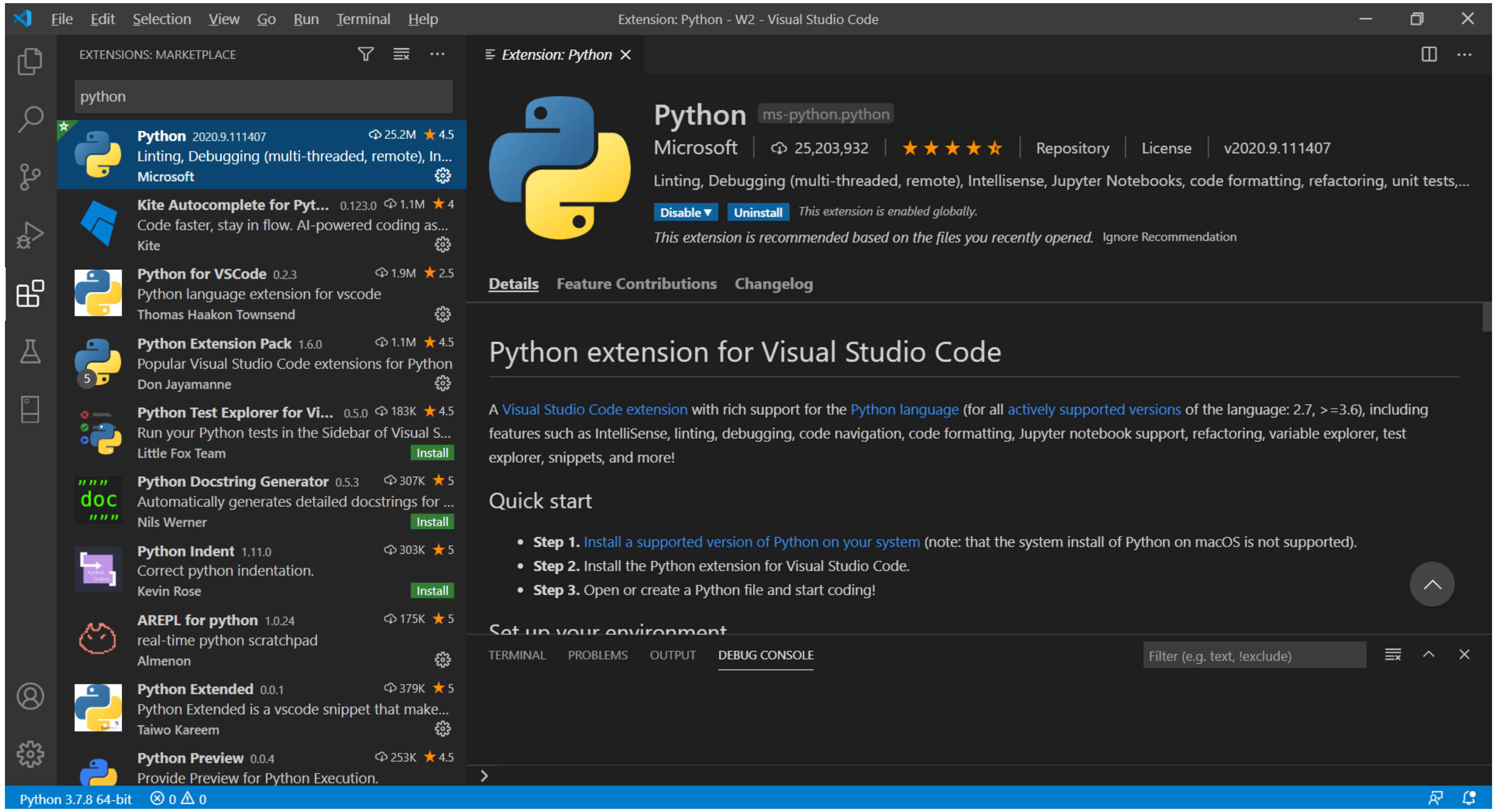
- Lecture notes are available in
- https://github.com/kkfyuen/PythonWorkshop2020

- The previous version of the slides can be found in
- https://github.com/kkfyuen/ANL251Python

Outline

1. Install Visual Studio Code
2. Boolean expressions
3. Conditional control flow
4. Use while-loop for repeated tasks
5. Lists and for-loop

Install Visual Studio Code and Extensions



python

Python 2020.9.111407 25.2M ★4.5
Linting, Debugging (multi-threaded, remote), In...
Microsoft

Kite Autocomplete for Pyt... 0.123.0 1.1M ★4
Code faster, stay in flow. AI-powered coding as...
Kite

Python for VSCode 0.2.3 1.9M ★2.5
Python language extension for vscode
Thomas Haakon Townsend

Python Extension Pack 1.6.0 1.1M ★4.5
Popular Visual Studio Code extensions for Python
Don Jayamanne

Python Test Explorer for Vi... 0.5.0 183K ★4.5
Run your Python tests in the Sidebar of Visual S...
Little Fox Team

Python Docstring Generator 0.5.3 307K ★5
Automatically generates detailed docstrings for ...
Nils Werner

Python Indent 1.11.0 303K ★5
Correct python indentation.
Kevin Rose

AREPL for python 1.0.24 175K ★5
real-time python scratchpad
Almenon

Python Extended 0.0.1 379K ★5
Python Extended is a vscode snippet that make...
Taiwo Kareem

Python Preview 0.0.4 253K ★4.5
Provide Preview for Python Execution.

Python ms-python.python
Microsoft | 25,203,932 | ★★★★★ | Repository | License | v2020.9.111407
Linting, Debugging (multi-threaded, remote), Intellisense, Jupyter Notebooks, code formatting, refactoring, unit tests,...
This extension is enabled globally.
This extension is recommended based on the files you recently opened. Ignore Recommendation

Python extension for Visual Studio Code

A Visual Studio Code extension with rich support for the Python language (for all actively supported versions of the language: 2.7, >=3.6), including features such as IntelliSense, linting, debugging, code navigation, code formatting, Jupyter notebook support, refactoring, variable explorer, test explorer, snippets, and more!

Quick start

- **Step 1.** Install a supported version of Python on your system (note: that the system install of Python on macOS is not supported).
- **Step 2.** Install the Python extension for Visual Studio Code.
- **Step 3.** Open or create a Python file and start coding!

Set up your environment

Python 3.7.8 64-bit ⊗ 0 ⚠ 0

Download and install

https://code.visualstudio.com/

← After install Visual Code Search "Python" to install extensions

Use command palette to select your Python interpreter version
Python: Select Interpreter

https://code.visualstudio.com/docs/python/environments

Some basic knowledge

Designing a Program

- Programs must be designed before they are written
- Program development cycle:
  - Design the program
  - Write the code
  - Correct syntax errors
  - Test the program
  - Correct logic errors

(Toby, 2021)

Designing a Program (cont'd.)

- Design is the most important part of the program development cycle
- Understand the task that the program is to perform
  - Work with customer to get a sense what the program is supposed to do
  - Ask questions about program details
  - Create one or more software requirements

(Toby, 2021)

Designing a Program (cont'd.)

- Determine the steps that must be taken to perform the task
  - Break down required task into a series of steps
  - Create an algorithm, listing logical steps that must be taken
- Algorithm: set of well-defined logical steps that must be taken to perform a task

(Toby, 2021)

Pseudocode

- Pseudocode: fake code
  - Informal language that has no syntax rule
  - Not meant to be compiled or executed
  - Used to create model program
    - No need to worry about syntax errors, can focus on program’s design
    - Can be translated directly into actual code in any programming language

(Toby, 2021)

Flowcharts

- Flowchart: diagram that graphically depicts the steps in a program
  - Ovals are terminal symbols
  - Parallelograms are input and output symbols
  - Rectangles are processing symbols
  - Symbols are connected by arrows that represent the flow of the program

(Toby, 2021)

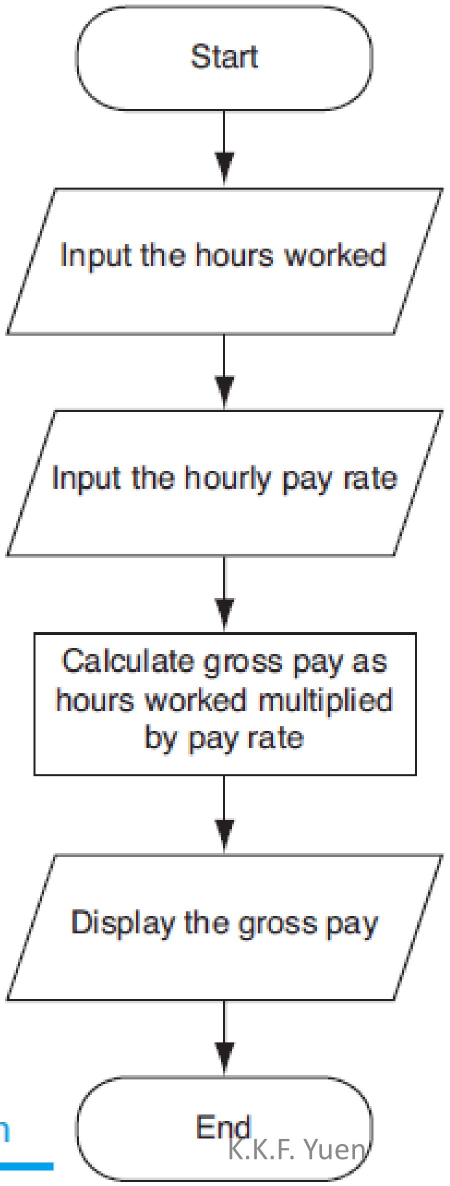

Start
Input the hours worked
Input the hourly pay rate
Calculate gross pay as hours worked multiplied by pay rate
Display the gross pay
End

Figure 2-2 Flowchart for the pay calculating program

(Toby, 2021)

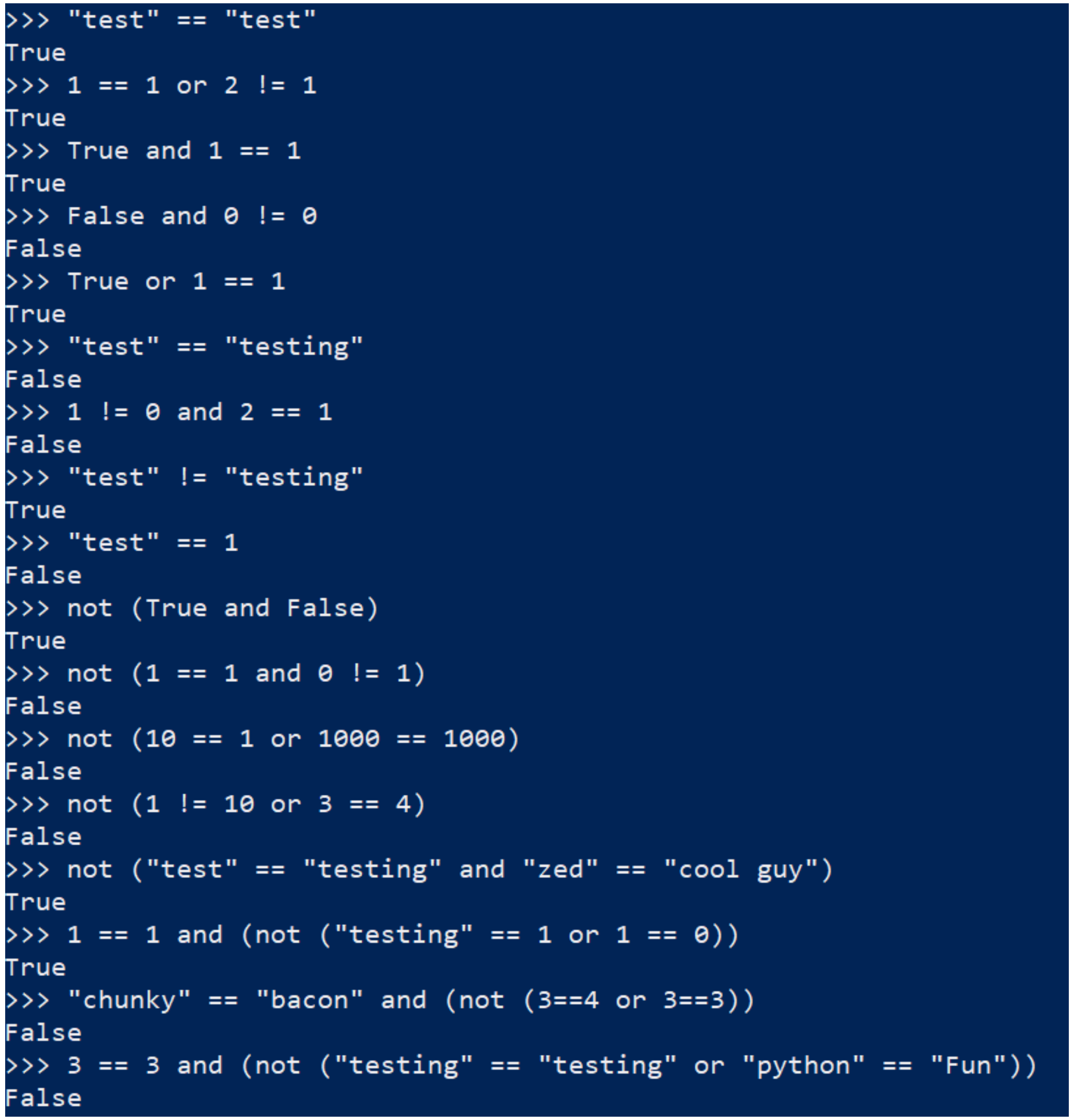
Boolean Expressions

Logical Operators

- Logical operators: operators that can be used to create complex Boolean expressions
  - `and` operator and `or` operator: binary operators, connect two Boolean expressions into a compound Boolean expression
  - `not` operator: unary operator, reverses the truth of its Boolean operand

Expression	Value of the Expression
False and False	False
False and True	False
True and False	False
True and True	True

Expression	Value of the Expression
False or False	False
False or True	True
True or False	True
True or True	True

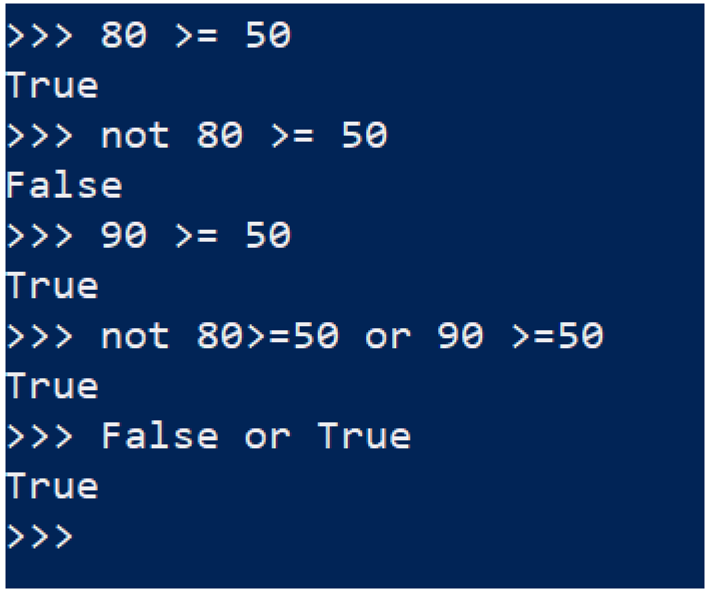


```
>>> "test" == "test"
True
>>> 1 == 1 or 2 != 1
True
>>> True and 1 == 1
True
>>> False and 0 != 0
False
>>> True or 1 == 1
True
>>> "test" == "testing"
False
>>> 1 != 0 and 2 == 1
False
>>> "test" != "testing"
True
>>> "test" == 1
False
>>> not (True and False)
True
>>> not (1 == 1 and 0 != 1)
False
>>> not (10 == 1 or 1000 == 1000)
False
>>> not (1 != 10 or 3 == 4)
False
>>> not ("test" == "testing" and "zed" == "cool guy")
True
>>> 1 == 1 and (not ("testing" == 1 or 1 == 0))
True
>>> "chunky" == "bacon" and (not (3==4 or 3==3))
False
>>> 3 == 3 and (not ("testing" == "testing" or "python" == "Fun"))
False
```

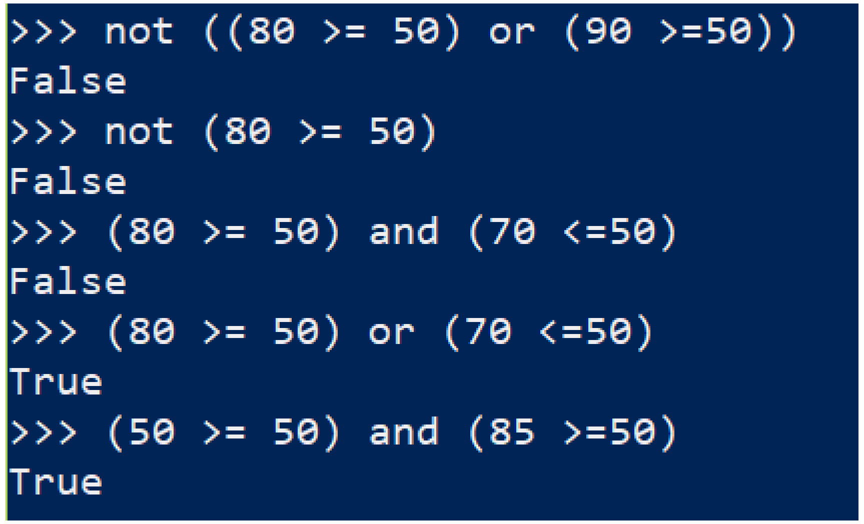
Ex28.py

Quiz 1

Expression
not 80 >= 50 or 90 >= 50
not ((80 >= 50) or (90 >= 50))
not (80 >= 50)
(80 >= 50) and (70 <= 50)
(80 >= 50) or (70 <= 50)
50 >= 50 and 85 >= 50



```
>>> 80 >= 50
True
>>> not 80 >= 50
False
>>> 90 >= 50
True
>>> not 80>=50 or 90 >=50
True
>>> False or True
True
>>>
```



```
>>> not ((80 >= 50) or (90 >=50))
False
>>> not (80 >= 50)
False
>>> (80 >= 50) and (70 <=50)
False
>>> (80 >= 50) or (70 <=50)
True
>>> (50 >= 50) and (85 >=50)
True
```

Discussion 1

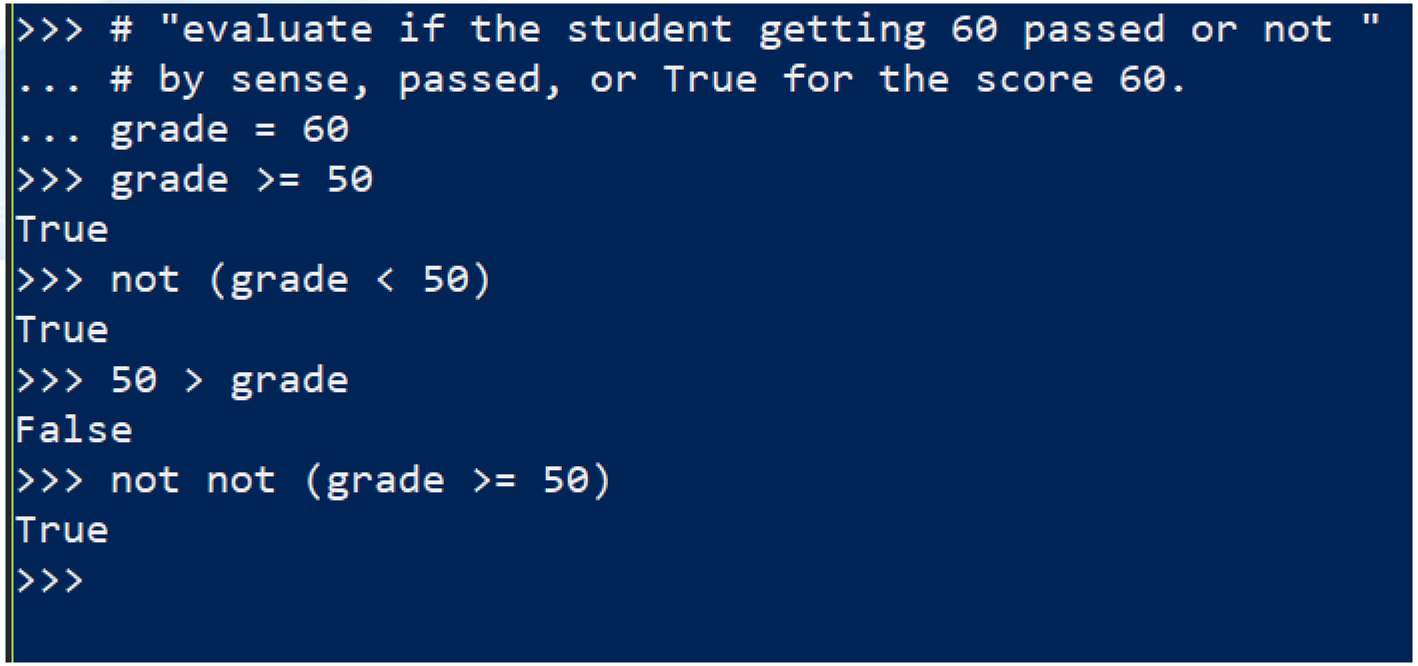
The minimum passing grade is 50. Variable grade refers to the grade for a student. Do the expressions below correctly represent the English sentence: "The student passed."?

grade >= 50

not (grade < 50)

50 > grade

not not (grade >= 50)



```
>>> # "evaluate if the student getting 60 passed or not "
... # by sense, passed, or True for the score 60.
... grade = 60
>>> grade >= 50
True
>>> not (grade < 50)
True
>>> 50 > grade
False
>>> not not (grade >= 50)
True
>>>
```

Discussion 2

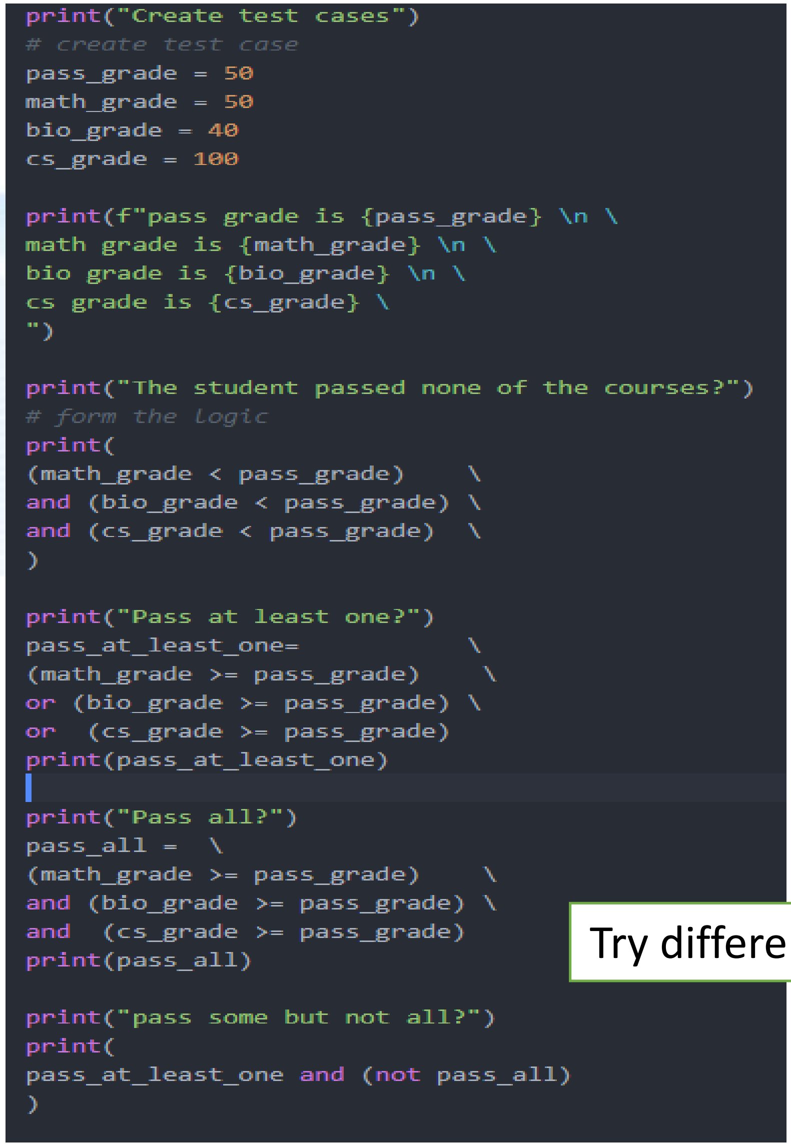
The minimum passing grade is 50. The variables math_grade, bio_grade, and cs_grade represent a student's final grades in three courses. Write the boolean expressions to correctly represent each of the English sentences below:

The student passed none of the courses

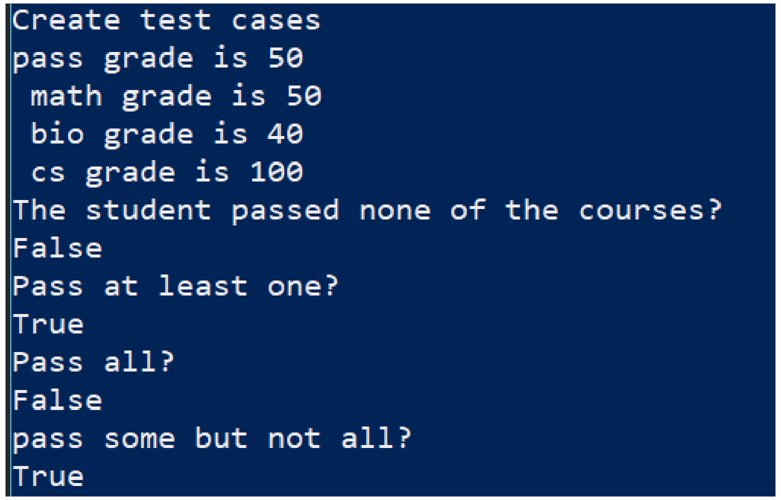
The student passed at least one of the courses.

The student passed all of the courses.

The student passed some but not all of the courses



```
print("Create test cases")
# create test case
pass_grade = 50
math_grade = 50
bio_grade = 40
cs_grade = 100

print(f"pass grade is {pass_grade} \n \
math grade is {math_grade} \n \
bio grade is {bio_grade} \n \
cs grade is {cs_grade} \
")

print("The student passed none of the courses?")
# form the logic
print(
(math_grade < pass_grade)     \
and (bio_grade < pass_grade) \
and (cs_grade < pass_grade)  \
)

print("Pass at least one?")
pass_at_least_one=             \
(math_grade >= pass_grade)     \
or (bio_grade >= pass_grade) \
or  (cs_grade >= pass_grade)
print(pass_at_least_one)

print("Pass all?")
pass_all =  \
(math_grade >= pass_grade)     \
and (bio_grade >= pass_grade) \
and  (cs_grade >= pass_grade)
print(pass_all)

print("pass some but not all?")
print(
pass_at_least_one and (not pass_all)
)
```



```
Create test cases
pass grade is 50
 math grade is 50
 bio grade is 40
 cs grade is 100
The student passed none of the courses?
False
Pass at least one?
True
Pass all?
False
pass some but not all?
True
```

Try different inputs to test

Breaking Long Statements into Multiple Lines

- Long statements cannot be viewed on screen without scrolling and cannot be printed without cutting off
- Multiline continuation character (\\): Allows to break a statement into multiple lines
    - Example:

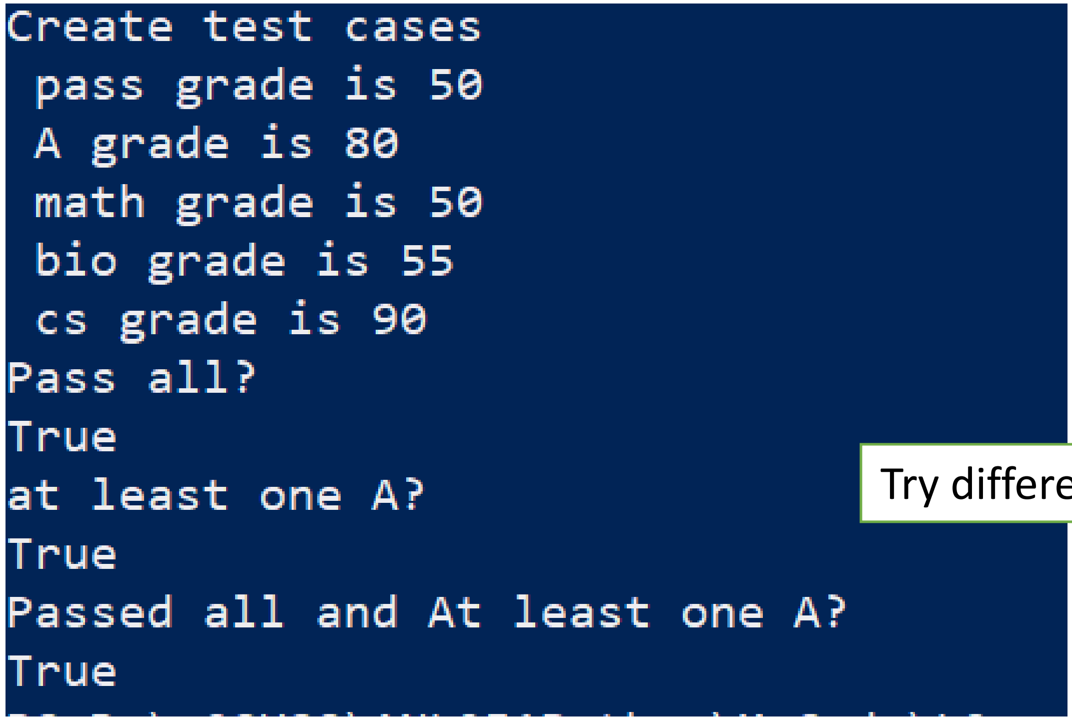
```
print('my first name is',\
                first_name)
```

(Toby , 2021)

Discussion 3

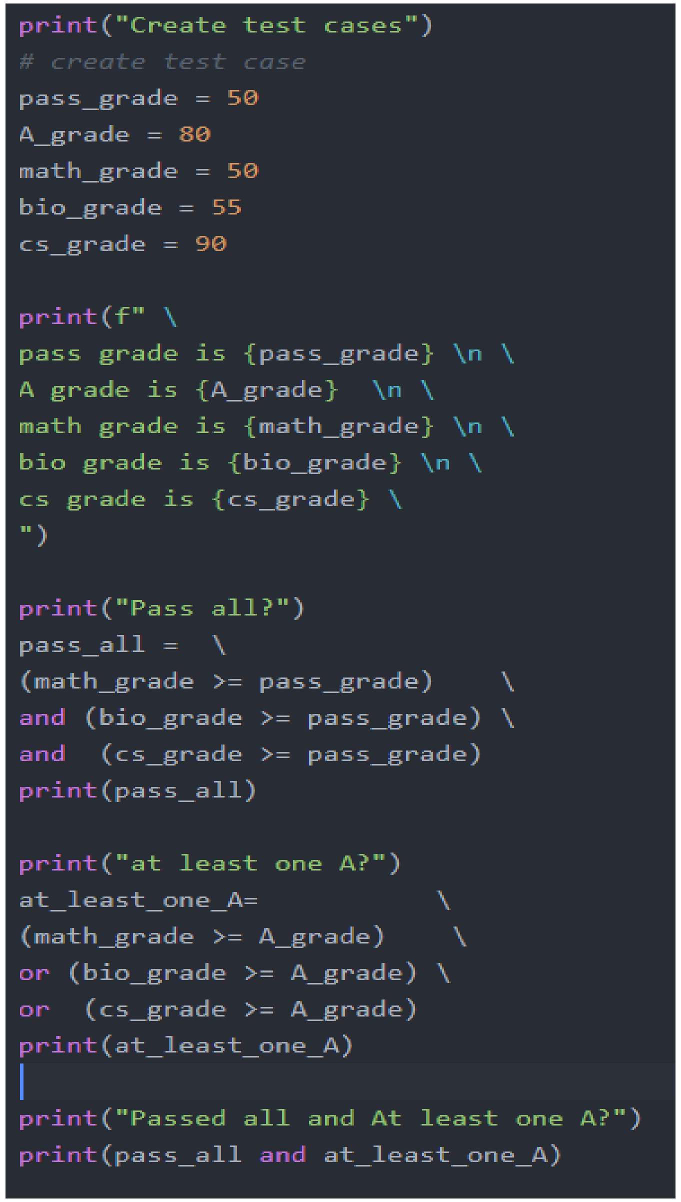
The minimum passing grade is 50, the minimum grade for "A" is 80 and variables math_grade, bio_grade, and cs_grade represent a student's final grades in three courses. Write the expression that represents the English sentence:

The student passed all their courses and earned at least one A.



```
Create test cases
 pass grade is 50
 A grade is 80
 math grade is 50
 bio grade is 55
 cs grade is 90
Pass all?
True
at least one A?
True
Passed all and At least one A?
True
```

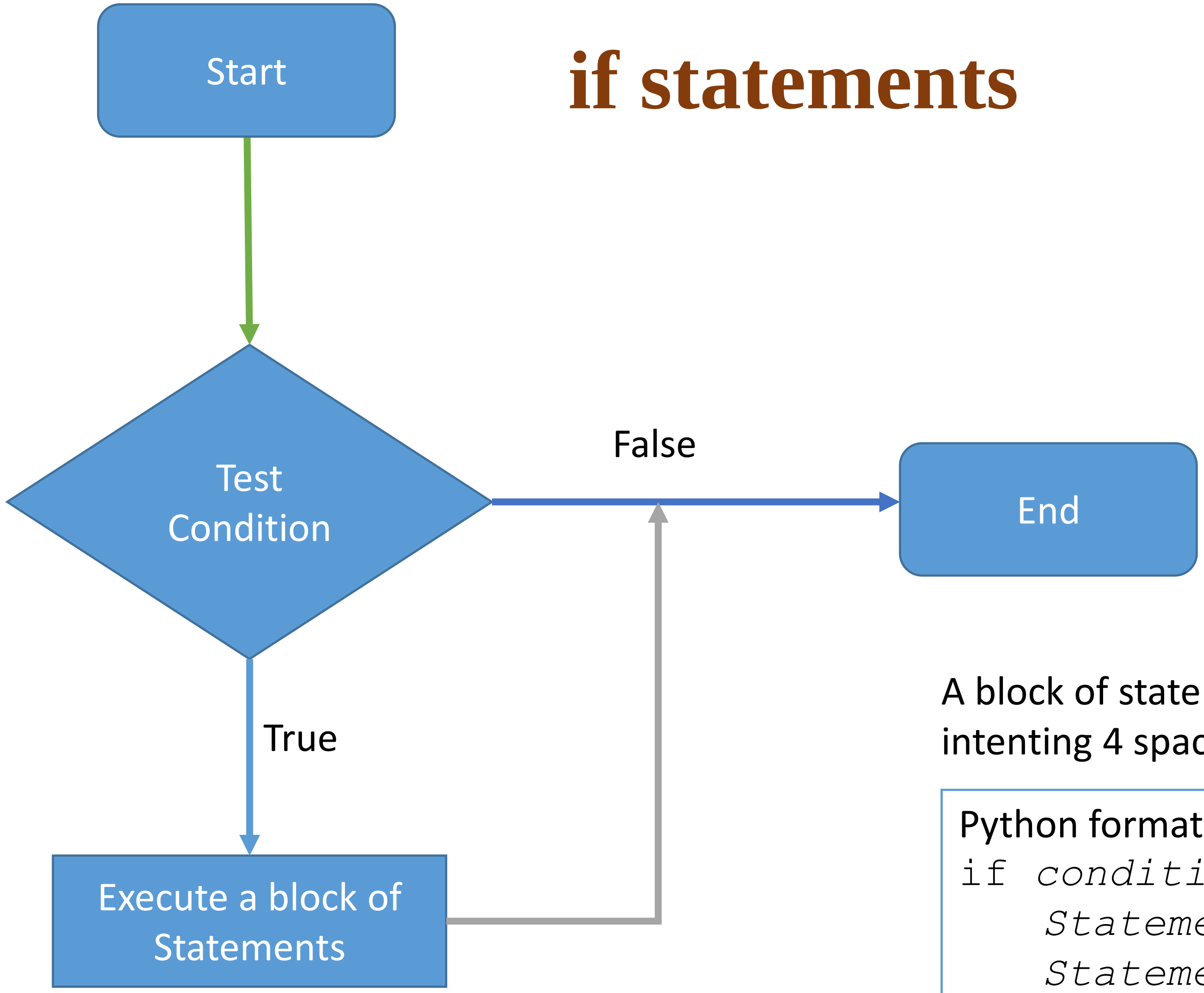
Try different inputs to test



```python
print("Create test cases")
# create test case
pass_grade = 50
A_grade = 80
math_grade = 50
bio_grade = 55
cs_grade = 90

print(f" \
pass grade is {pass_grade} \n \
A grade is {A_grade}  \n \
math grade is {math_grade} \n \
bio grade is {bio_grade} \n \
cs grade is {cs_grade} \
")

print("Pass all?")
pass_all =  \
(math_grade >= pass_grade)    \
and (bio_grade >= pass_grade) \
and  (cs_grade >= pass_grade)
print(pass_all)

print("at least one A?")
at_least_one_A=              \
(math_grade >= A_grade)     \
or (bio_grade >= A_grade) \
or  (cs_grade >= A_grade)
print(at_least_one_A)

print("Passed all and At least one A?")
print(pass_all and at_least_one_A)
```

Conditional Statements

if statements

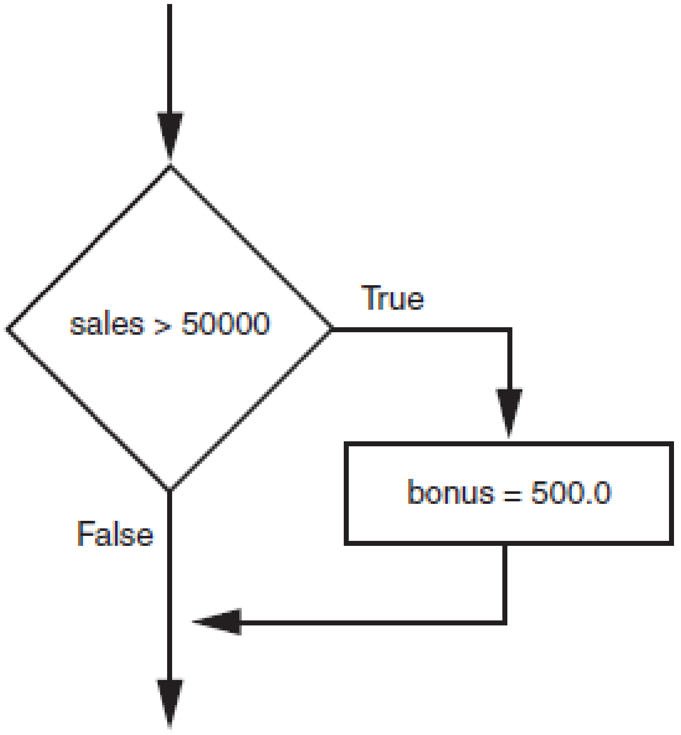


A block of statements is defined by intenting 4 spaces

Python format:
```
if condition:
    Statement
    Statement
```

Example

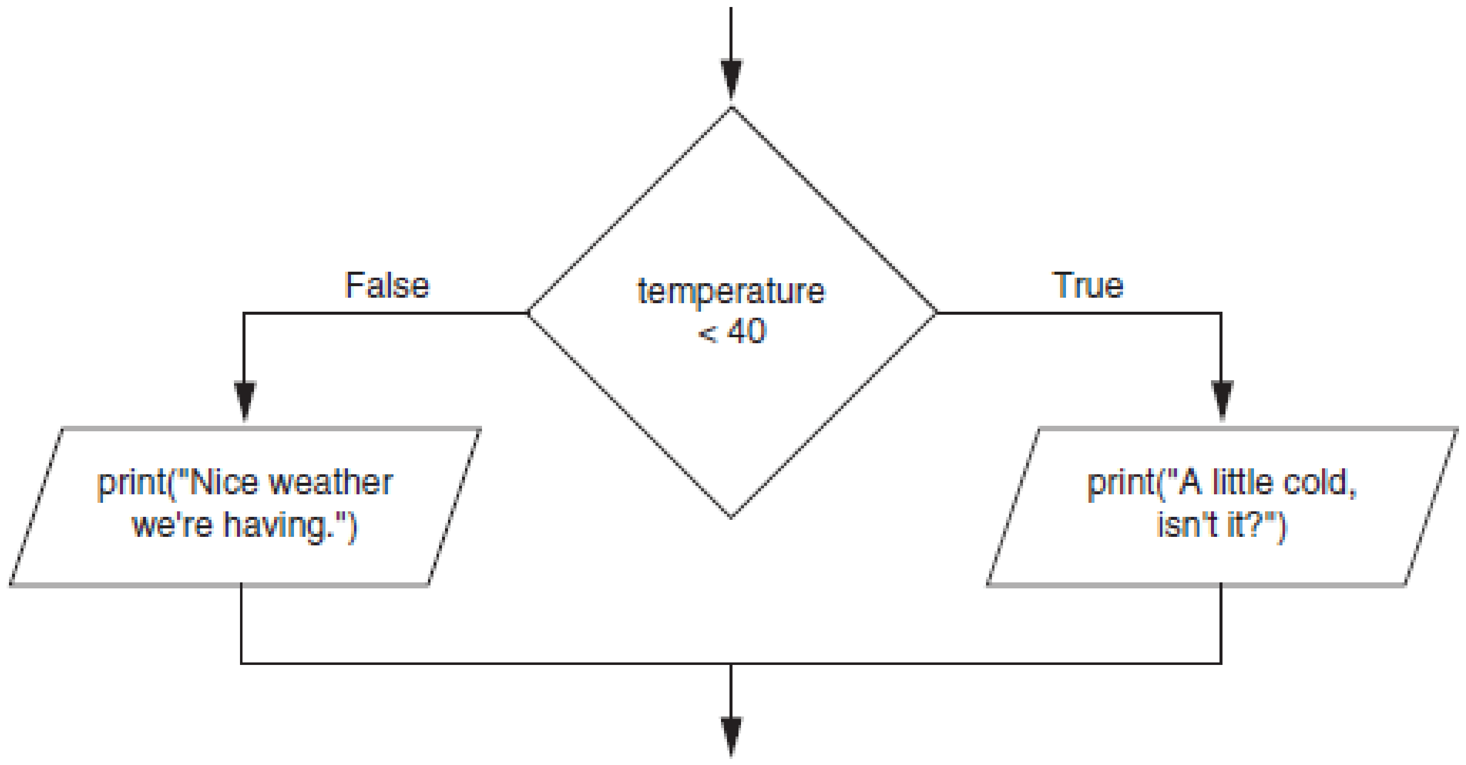
Figure 3-3 Example decision structure



(Toby , 2021)

The `if-else` Statement (cont'd.)

Figure 3-5 A dual alternative decision structure


temperature < 40
False
True
print("Nice weather we're having.")
print("A little cold, isn't it?")

(Toby , 2021)

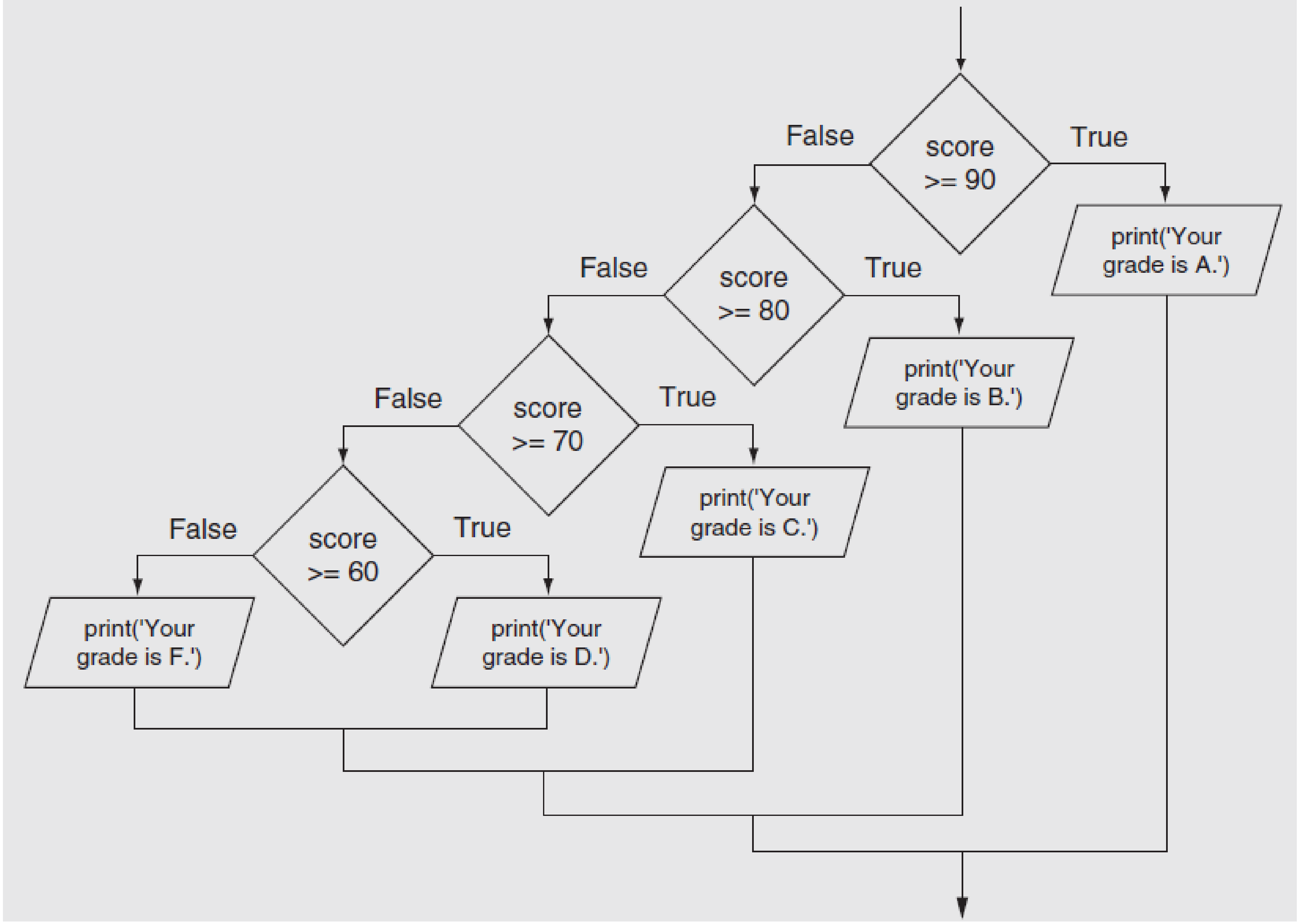
The `if-elif-else` Statement

- `if-elif-else` statement: special version of a decision structure
  - Makes logic of nested decision structures simpler to write
    - Can include multiple `elif` statements
  - Syntax:

```
if condition1
        statements
elif condition2
        statements
else
        statements
```

(Toby , 2021)

Figure 3-15 Nested decision structure to determine a grade


score >= 90
True
False
print('Your grade is A.')
score >= 80
True
False
print('Your grade is B.')
score >= 70
True
False
print('Your grade is C.')
score >= 60
True
False
print('Your grade is D.')
print('Your grade is F.')

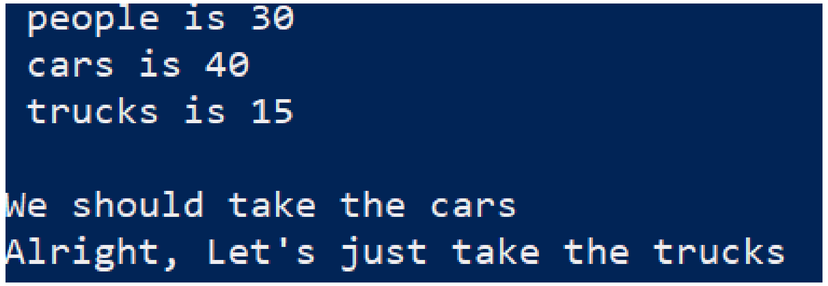
(Toby , 2021)

Discussion 4

Conditional statements

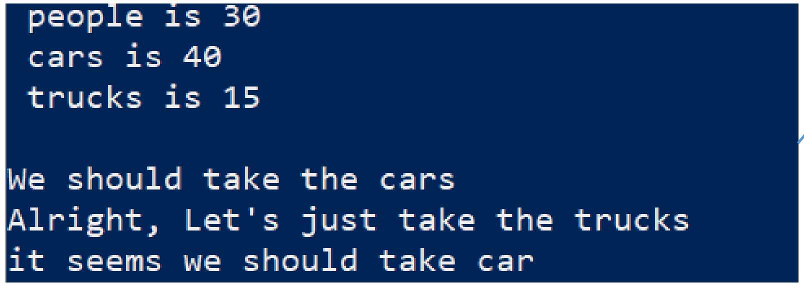
1. Change the numbers of cars, people, and trucks, and then trace through each if-statement to see what will be printed.
2. Try some more complex Boolean expressions like cars > people or trucks < cars.

Original

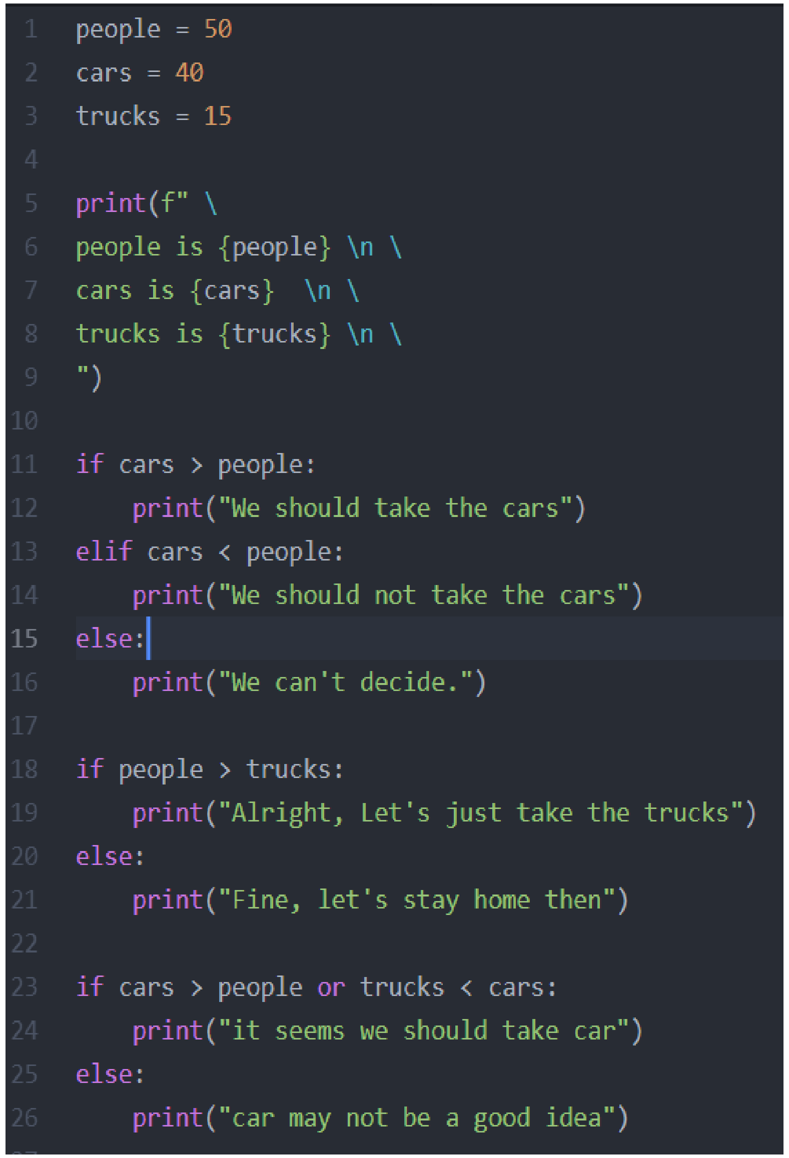


```
people is 30
cars is 40
trucks is 15

We should take the cars
Alright, Let's just take the trucks
```

Some test



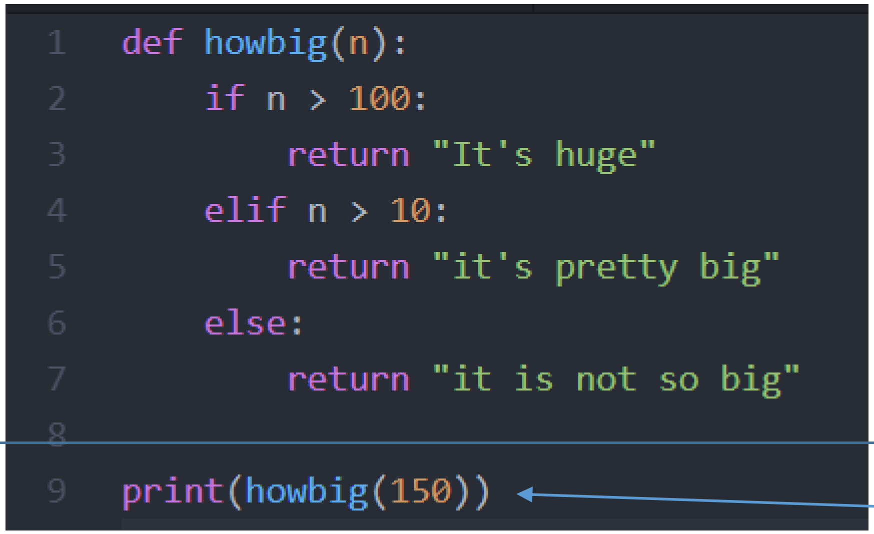
```
people is 30
cars is 40
trucks is 15

We should take the cars
Alright, Let's just take the trucks
it seems we should take car
```



```python
people = 50
cars = 40
trucks = 15

print(f" \
people is {people} \n \
cars is {cars}  \n \
trucks is {trucks} \n \
")

if cars > people:
    print("We should take the cars")
elif cars < people:
    print("We should not take the cars")
else:
    print("We can't decide.")

if people > trucks:
    print("Alright, Let's just take the trucks")
else:
    print("Fine, let's stay home then")

if cars > people or trucks < cars:
    print("it seems we should take car")
else:
    print("car may not be a good idea")
```

Quiz 2

What is the result for the code executed below.

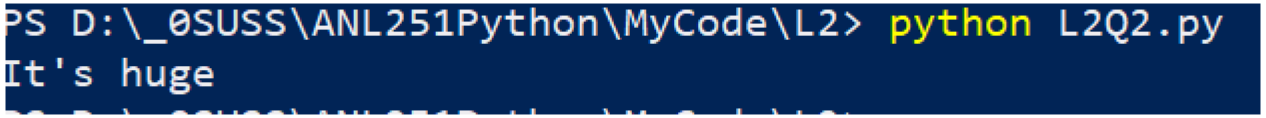


```
def howbig(n):
    if n > 100:
        return "It's huge"
    elif n > 10:
        return "it's pretty big"
    else:
        return "it is not so big"

print(howbig(150))
```

Create a function
- Use def to define a function.
- Indent the spaces for block of statement.

call a function created by us

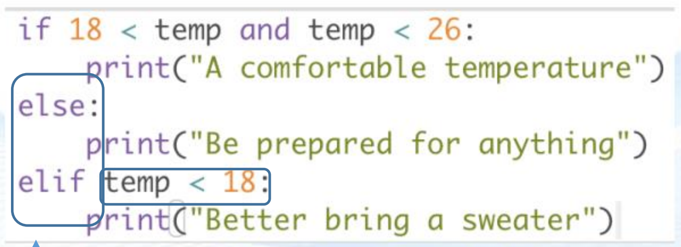
Output



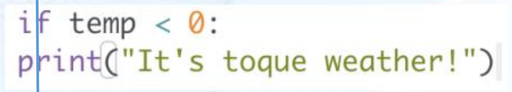
```
PS D:\_0SUSS\ANL251Python\MyCode\L2> python L2Q2.py
It's huge
```

Quiz 3

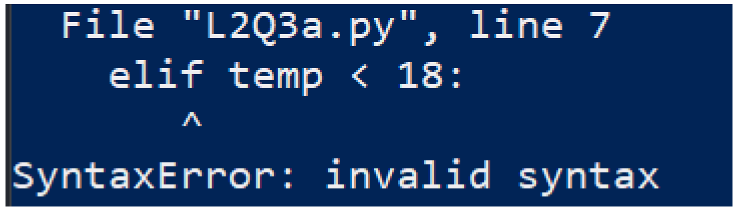
Correct the syntax error for each of the code below if there is any.



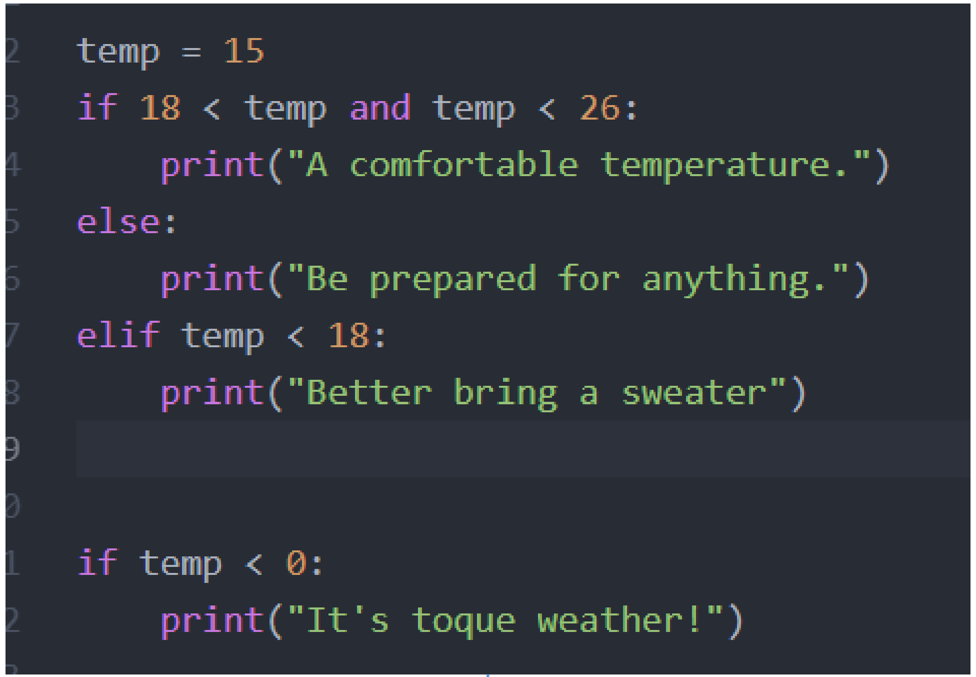
```
if 18 < temp and temp < 26:
    print("A comfortable temperature")
else:
    print("Be prepared for anything")
elif temp < 18:
    print("Better bring a sweater")
```



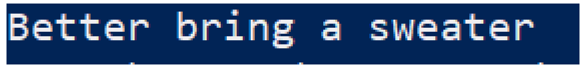
```
if temp < 0:
print("It's toque weather!")
```



```
  File "L2Q3a.py", line 7
    elif temp < 18:
       ^
SyntaxError: invalid syntax
```

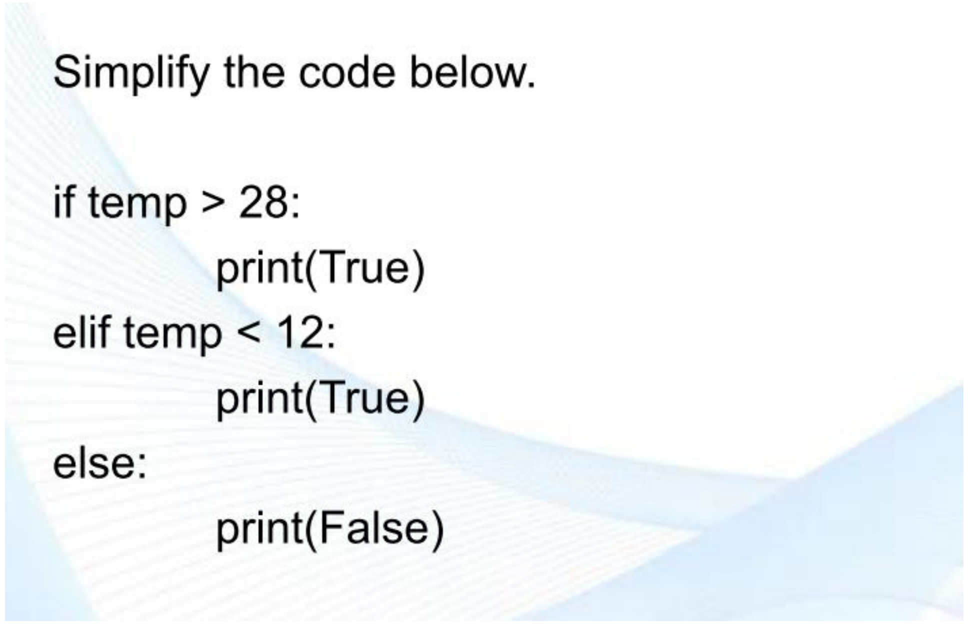


```
temp = 15
if 18 < temp and temp < 26:
    print("A comfortable temperature.")
else:
    print("Be prepared for anything.")
elif temp < 18:
    print("Better bring a sweater")


if temp < 0:
    print("It's toque weather!")
```



```
Better bring a sweater
```

Discussion 5

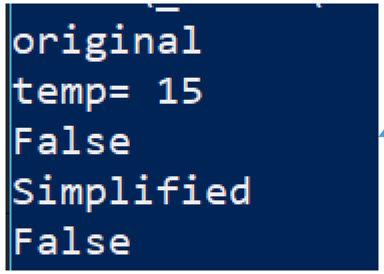


Simplify the code below.

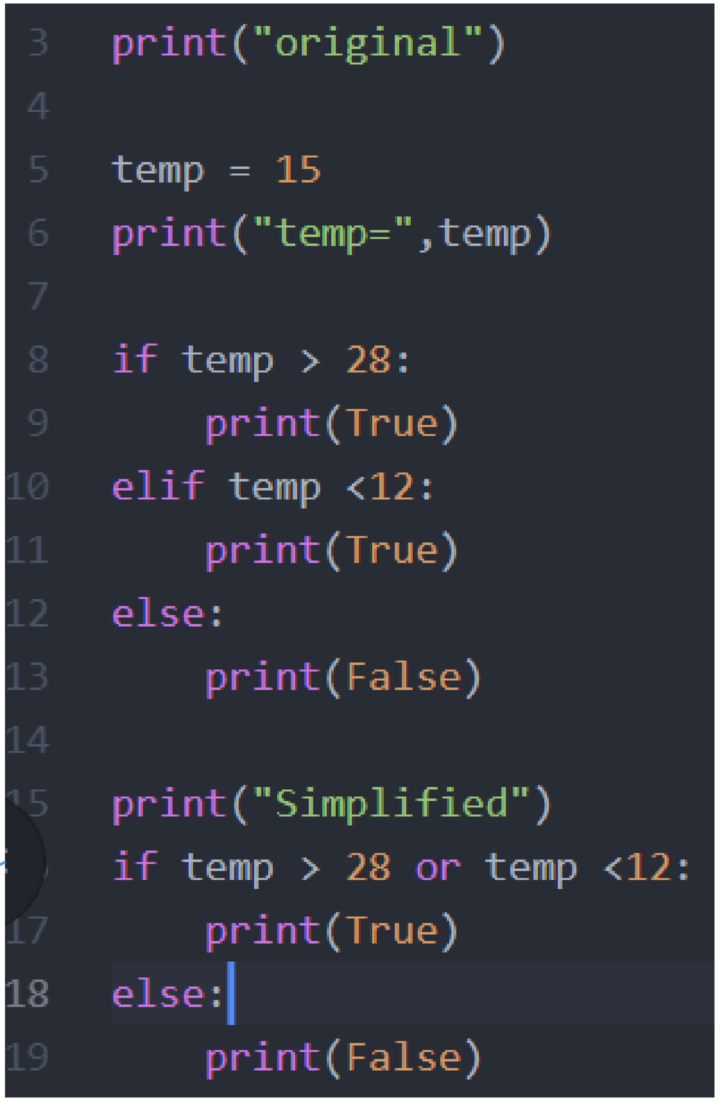
```
if temp > 28:
        print(True)
elif temp < 12:
        print(True)
else:
        print(False)
```

How to define test cases?

print



```
original
temp= 15
False
Simplified
False
```

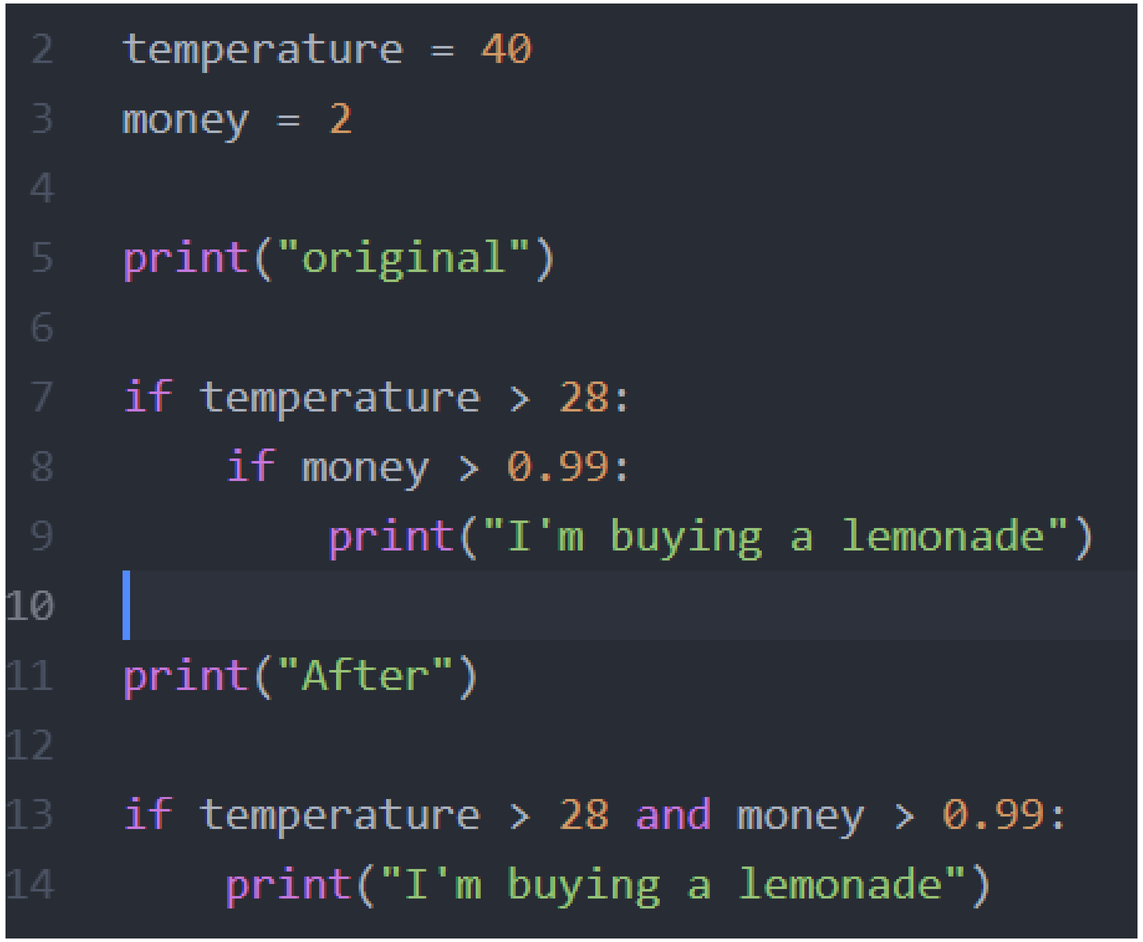


```
print("original")

temp = 15
print("temp=",temp)

if temp > 28:
    print(True)
elif temp <12:
    print(True)
else:
    print(False)

print("Simplified")
if temp > 28 or temp <12:
    print(True)
else:
    print(False)
```

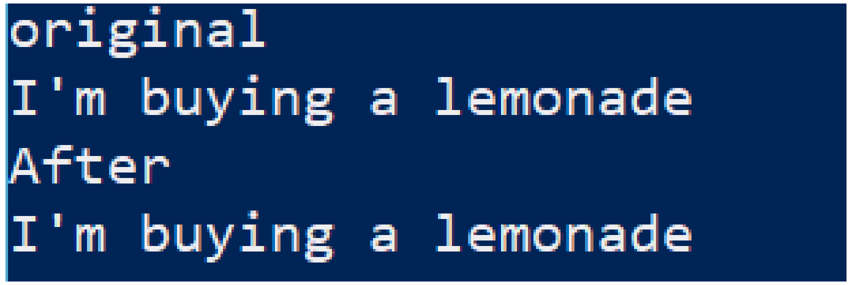
Discussion 6

Simplify the code below.

```
if temperature > 28:
        if money > 0.99:
                print("I'm buying a lemonade.")
```



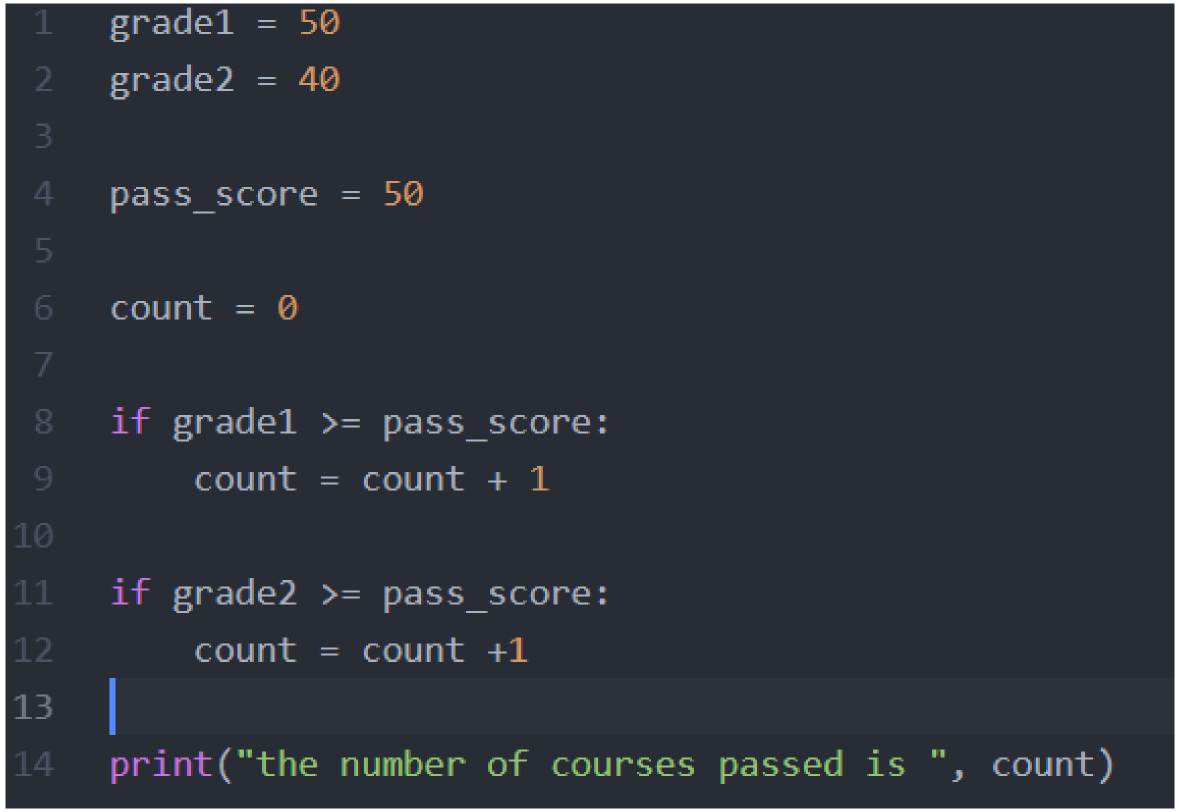
```
temperature = 40
money = 2

print("original")

if temperature > 28:
    if money > 0.99:
        print("I'm buying a lemonade")

print("After")

if temperature > 28 and money > 0.99:
    print("I'm buying a lemonade")
```



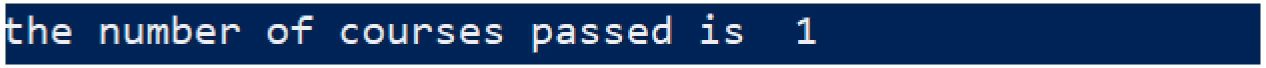
```
original
I'm buying a lemonade
After
I'm buying a lemonade
```

Quiz 4

Variables grade1 and grade2 represent grades in two courses. Write the code to count the number of courses passed (with a 50 or higher).



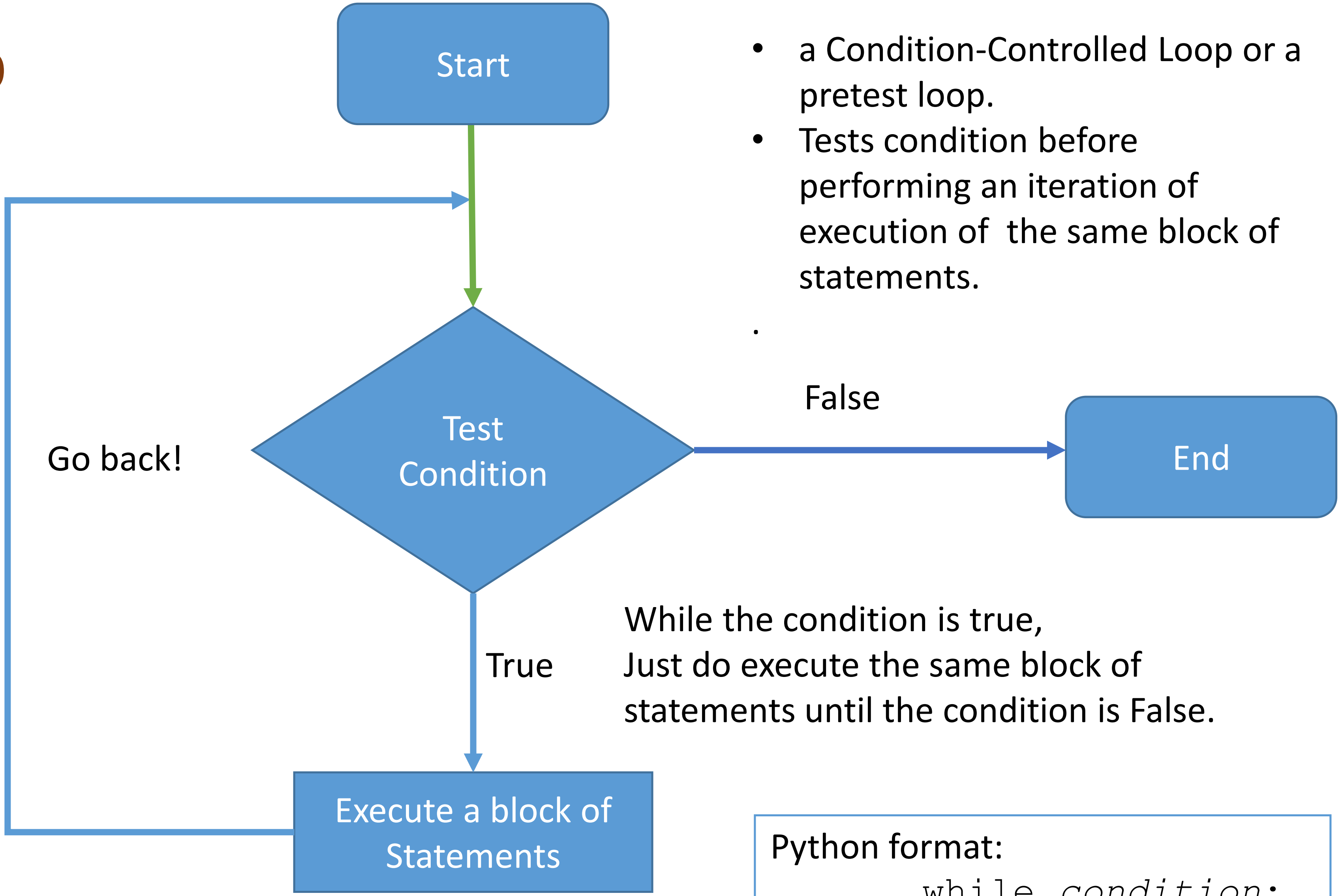
```python
grade1 = 50
grade2 = 40

pass_score = 50

count = 0

if grade1 >= pass_score:
    count = count + 1

if grade2 >= pass_score:
    count = count +1

print("the number of courses passed is ", count)
```



```
the number of courses passed is  1
```

While loop

while Loop



- a Condition-Controlled Loop or a pretest loop.
- Tests condition before performing an iteration of execution of the same block of statements.

.

While the condition is true,
Just do execute the same block of statements until the condition is False.

Python format:

```
        while condition:
                statements
```

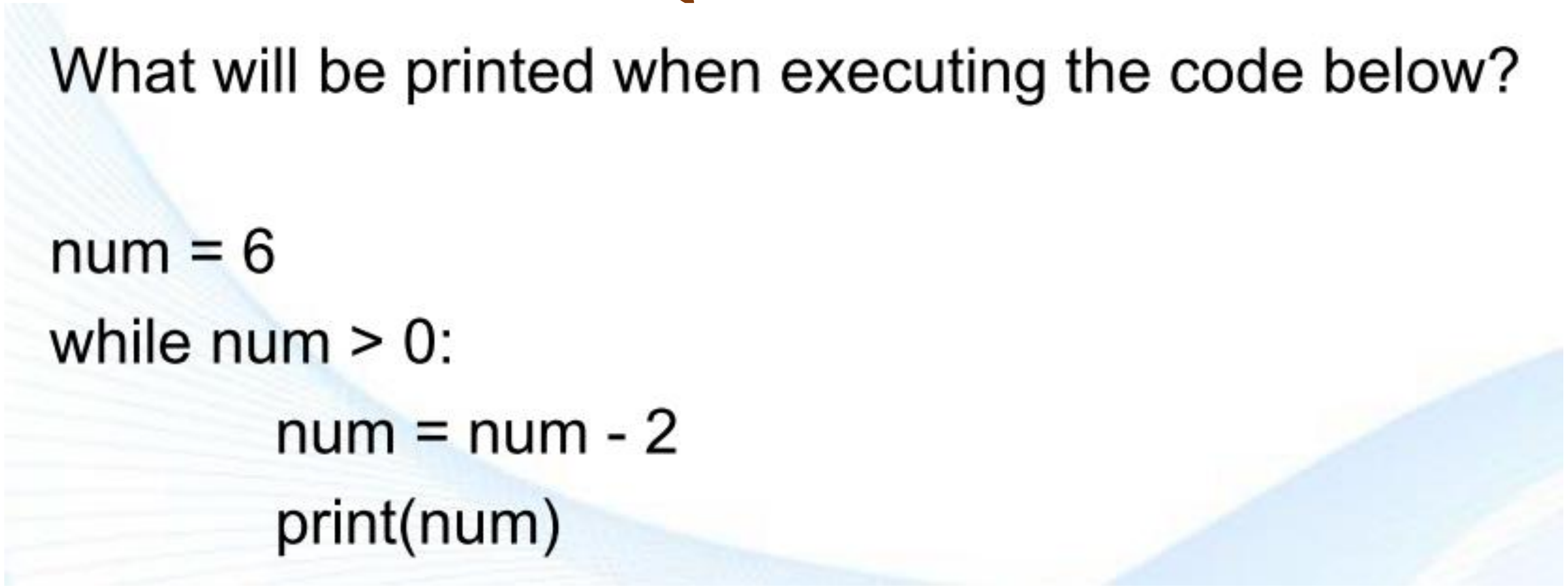
The **`while`** Loop: a Condition-Controlled Loop

- <u>`while` loop</u>: while condition is true, do something
  - Two parts:
    - Condition tested for true or false value
    - Statements repeated as long as condition is true
  - In flow chart, line goes back to previous part
  - General format:

```
while condition:
      statements
```

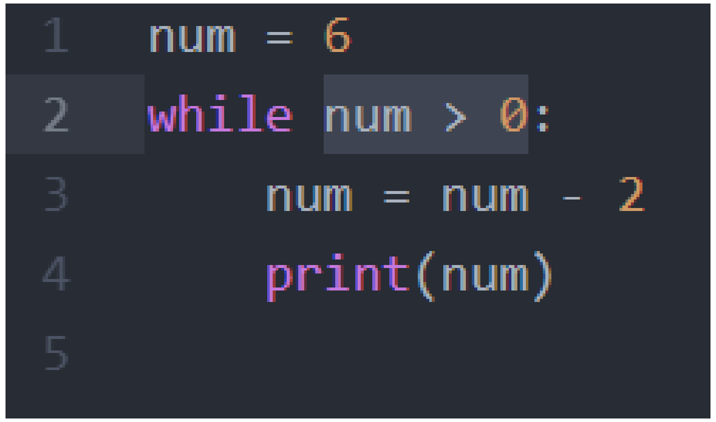
(Toby 2014)

Quiz 5

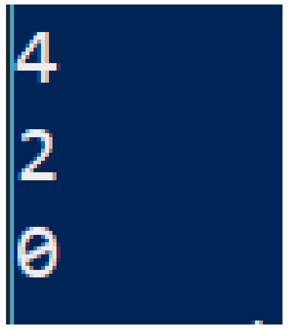


What will be printed when executing the code below?

```
num = 6
while num > 0:
        num = num - 2
        print(num)
```



```
num = 6
while num > 0:
    num = num - 2
    print(num)
```

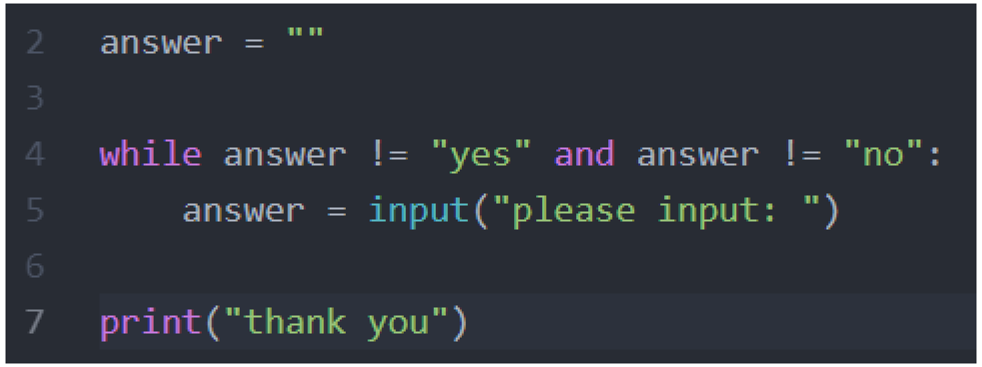


```
4
2
0
```

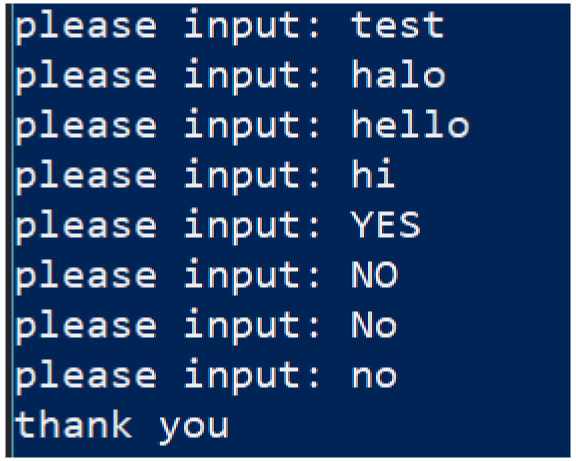
Press **Ctrl + C**
in case of infinite loop

Discussion 7

Write a program to ask the user for a "yes" or "no" input and continue asking until the user gives a valid response. Print the answer.



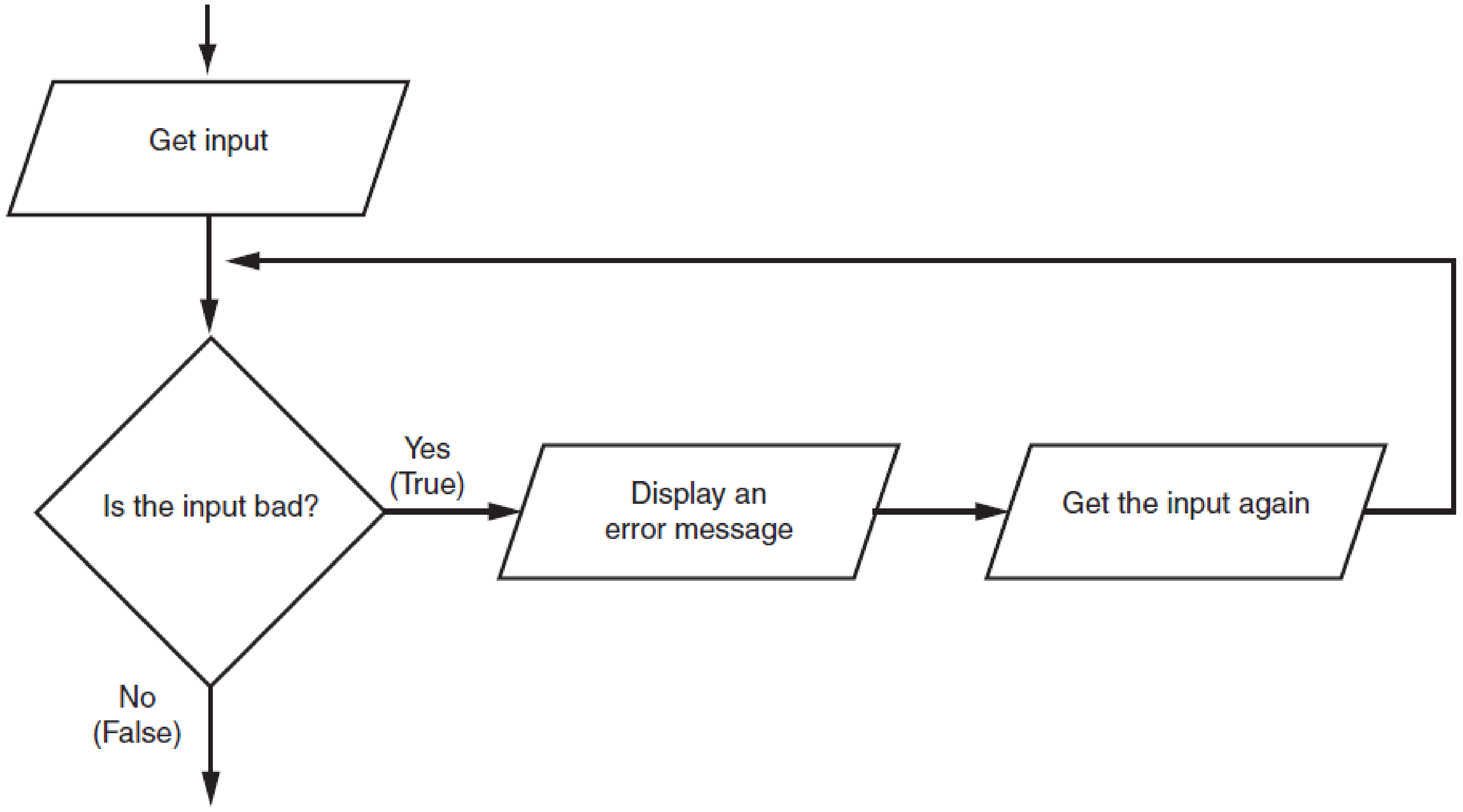
```python
answer = ""

while answer != "yes" and answer != "no":
    answer = input("please input: ")

print("thank you")
```



```
please input: test
please input: halo
please input: hello
please input: hi
please input: YES
please input: NO
please input: No
please input: no
thank you
```

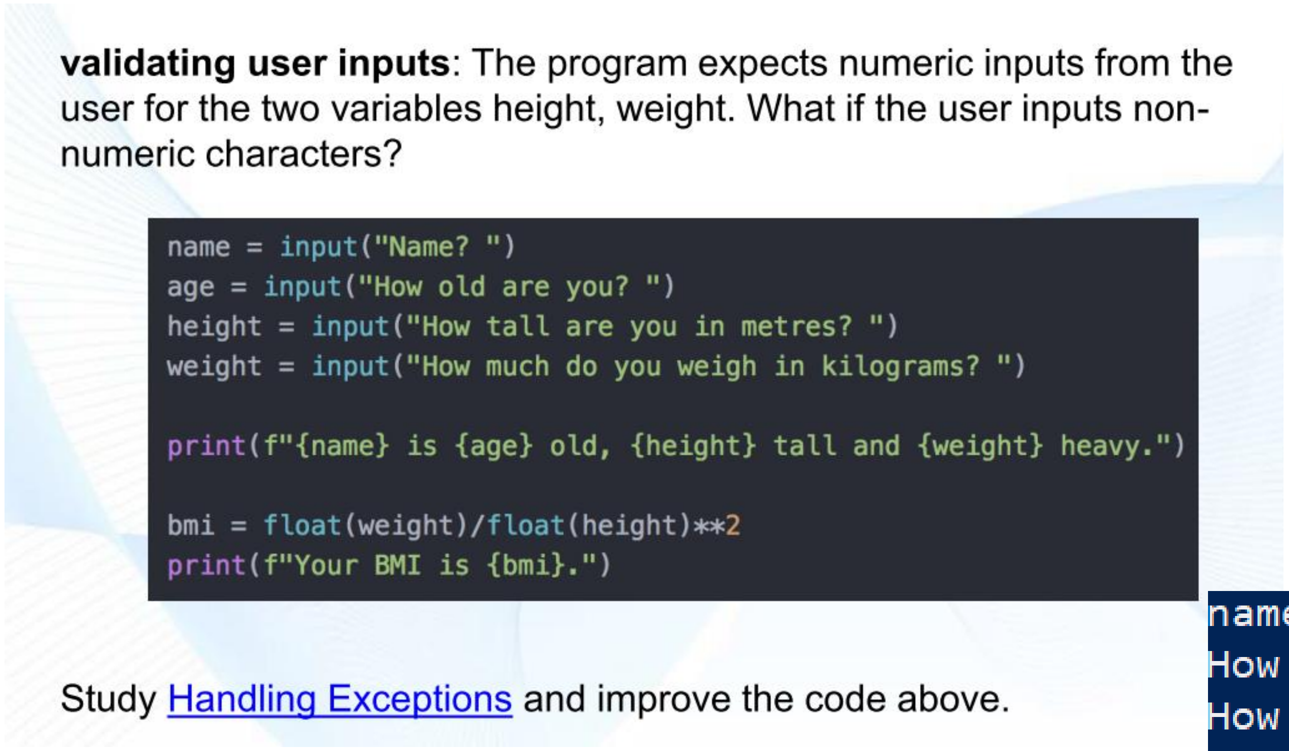
Input Validation Loops

Figure 4-7 Logic containing an input validation loop



(Toby , 2021)

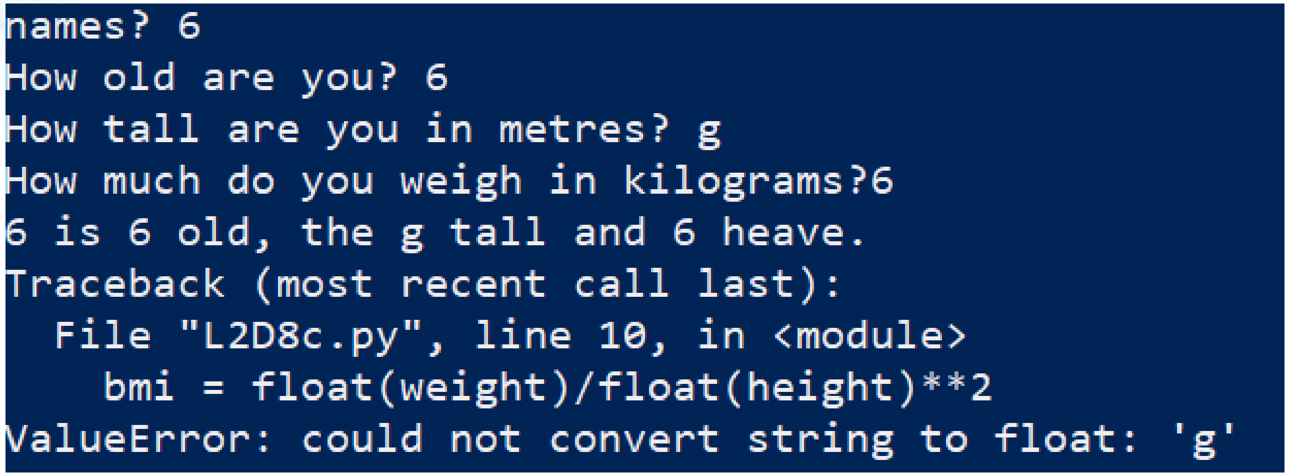
Discussion 8



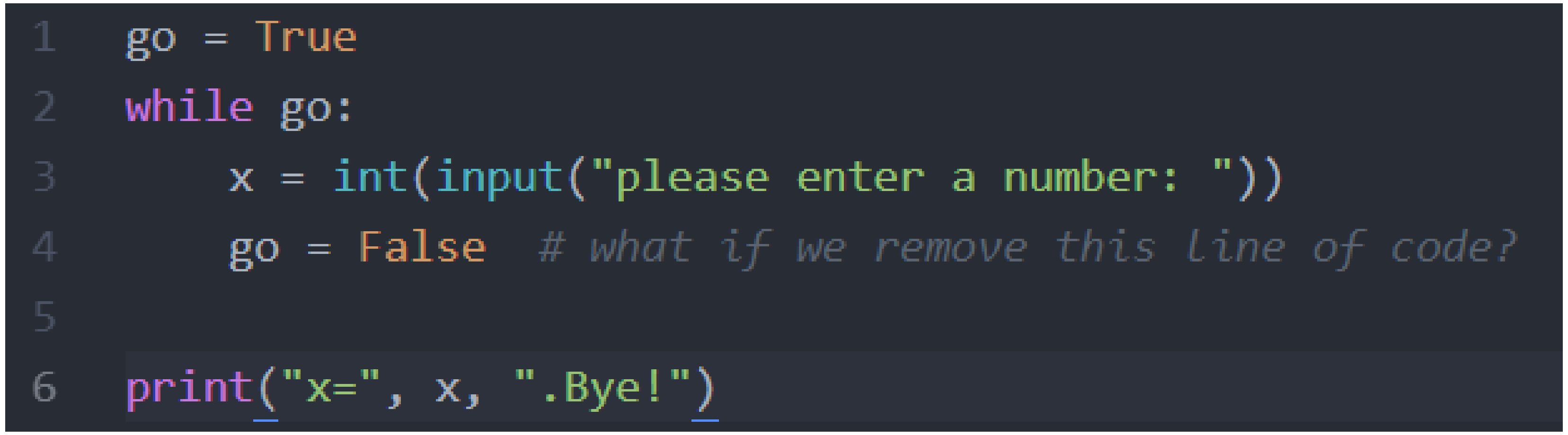
validating user inputs: The program expects numeric inputs from the user for the two variables height, weight. What if the user inputs non-numeric characters?

```
name = input("Name? ")
age = input("How old are you? ")
height = input("How tall are you in metres? ")
weight = input("How much do you weigh in kilograms? ")

print(f"{name} is {age} old, {height} tall and {weight} heavy.")

bmi = float(weight)/float(height)**2
print(f"Your BMI is {bmi}.")
```

Study Handling Exceptions and improve the code above.

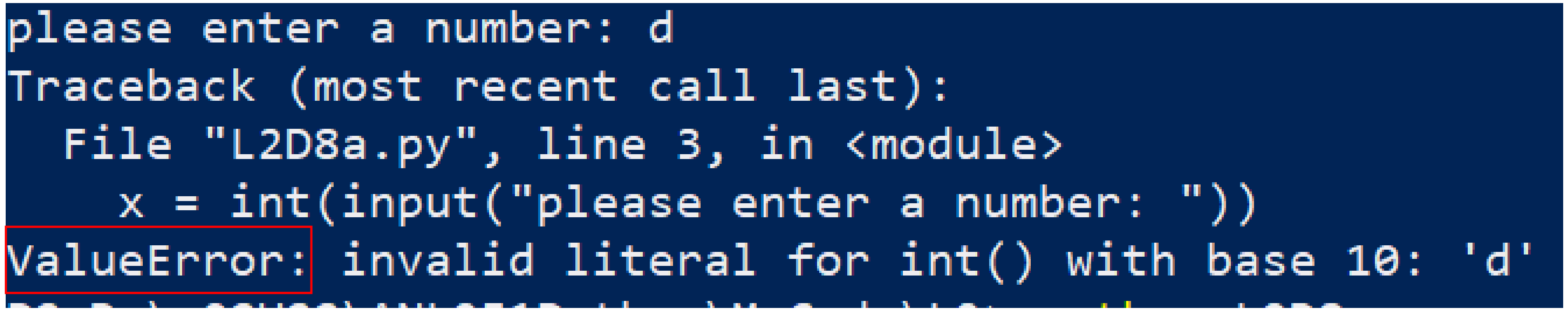


```
names? 6
How old are you? 6
How tall are you in metres? g
How much do you weigh in kilograms?6
6 is 6 old, the g tall and 6 heave.
Traceback (most recent call last):
  File "L2D8c.py", line 10, in <module>
    bmi = float(weight)/float(height)**2
ValueError: could not convert string to float: 'g'
```



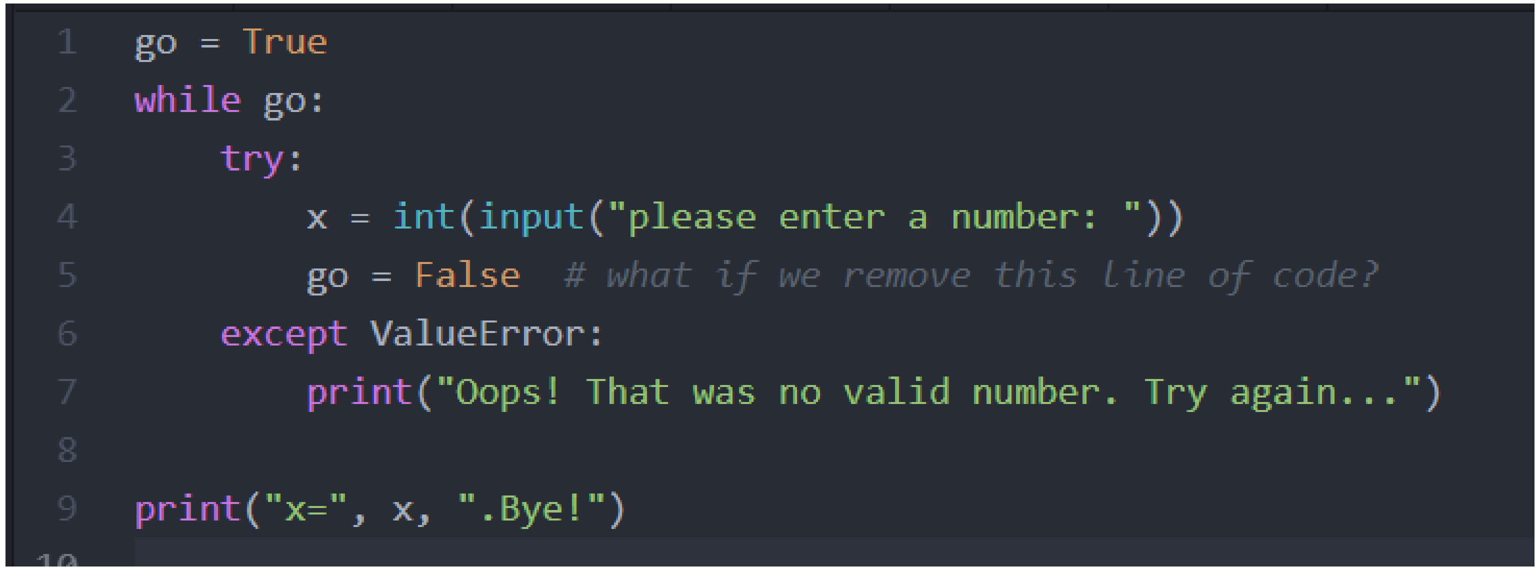
```
1  go = True
2  while go:
3      x = int(input("please enter a number: "))
4      go = False  # what if we remove this line of code?
5
6  print("x=", x, ".Bye!")
```

int() cannot convert a character into a number for some cases.

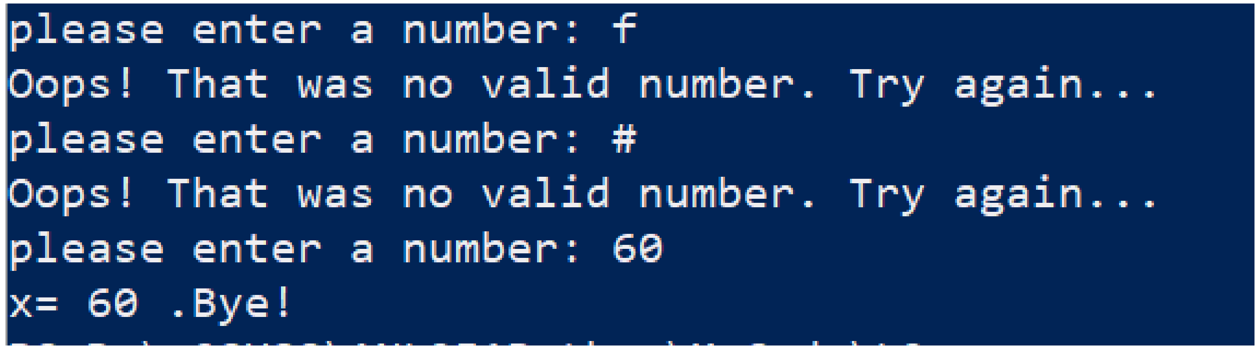


```
please enter a number: d
Traceback (most recent call last):
  File "L2D8a.py", line 3, in <module>
    x = int(input("please enter a number: "))
ValueError: invalid literal for int() with base 10: 'd'
```

Handling Exceptions



```python
go = True
while go:
    try:
        x = int(input("please enter a number: "))
        go = False  # what if we remove this line of code?
    except ValueError:
        print("Oops! That was no valid number. Try again...")

print("x=", x, ".Bye!")
```

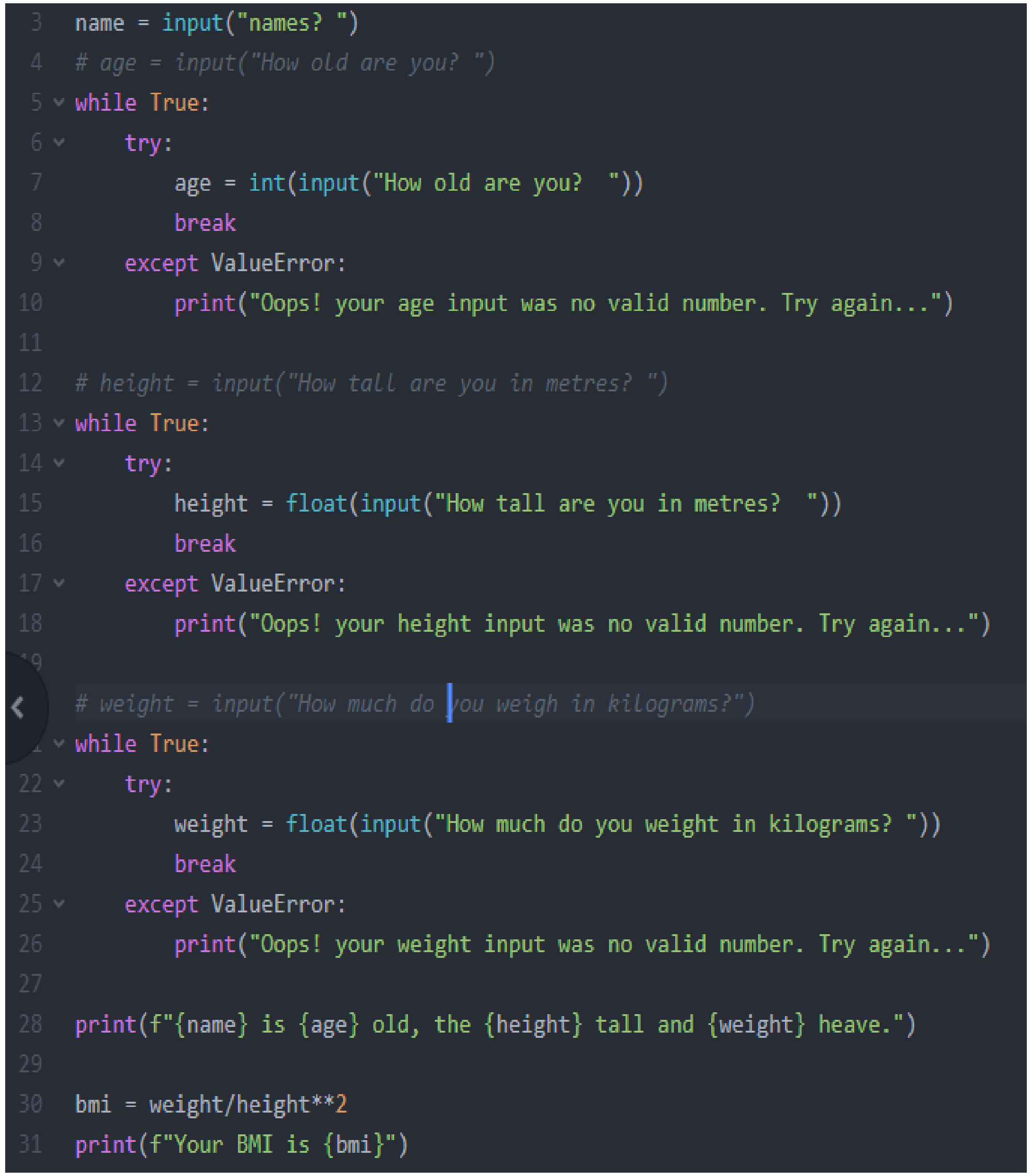


```
please enter a number: f
Oops! That was no valid number. Try again...
please enter a number: #
Oops! That was no valid number. Try again...
please enter a number: 60
x= 60 .Bye!
```

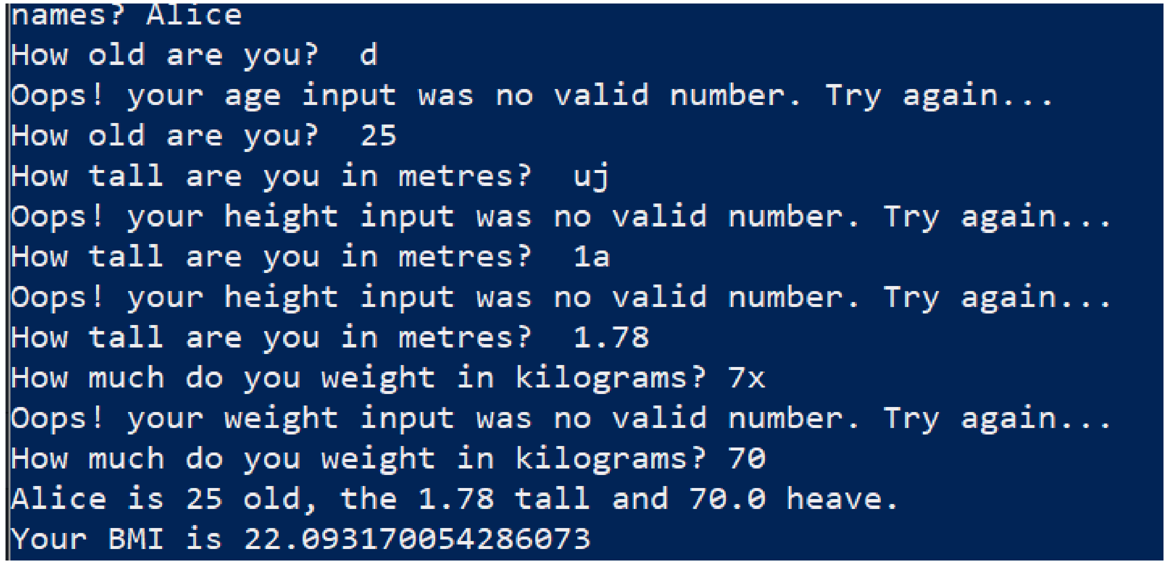
For more information…

https://docs.python.org/3/tutorial/errors.html#handling-exceptions

So modify the code for discussion 8



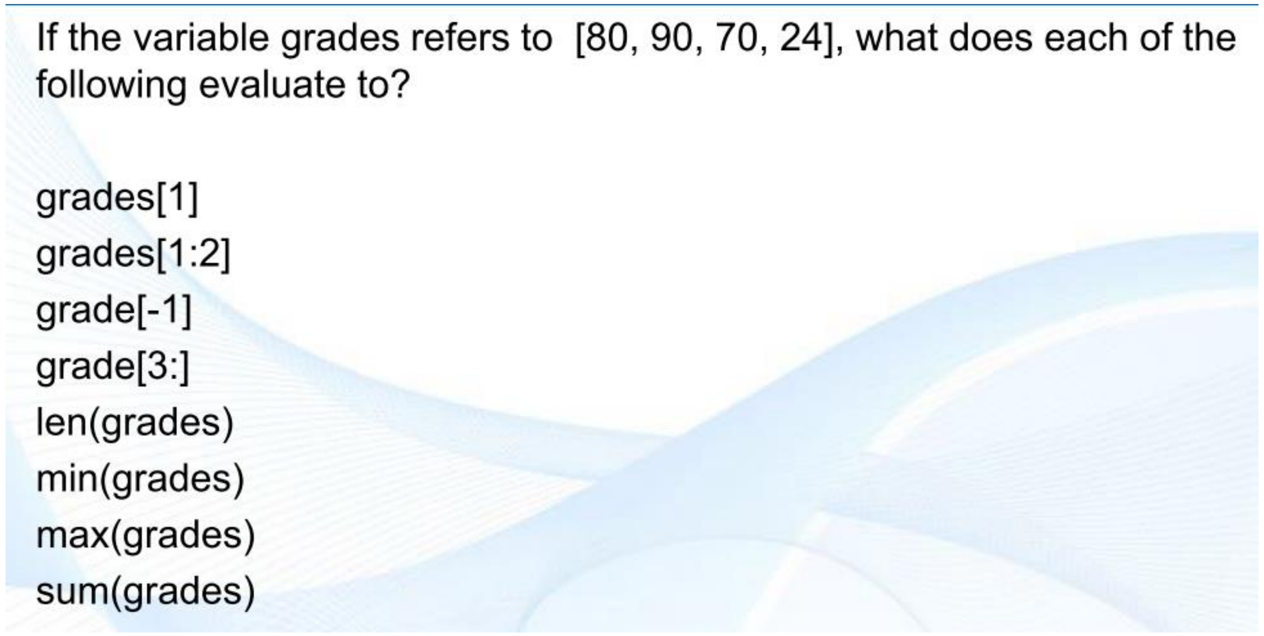
```
name = input("names? ")
# age = input("How old are you? ")
while True:
    try:
        age = int(input("How old are you?  "))
        break
    except ValueError:
        print("Oops! your age input was no valid number. Try again...")

# height = input("How tall are you in metres? ")
while True:
    try:
        height = float(input("How tall are you in metres?  "))
        break
    except ValueError:
        print("Oops! your height input was no valid number. Try again...")

# weight = input("How much do you weigh in kilograms?")
while True:
    try:
        weight = float(input("How much do you weight in kilograms? "))
        break
    except ValueError:
        print("Oops! your weight input was no valid number. Try again...")

print(f"{name} is {age} old, the {height} tall and {weight} heave.")

bmi = weight/height**2
print(f"Your BMI is {bmi}")
```



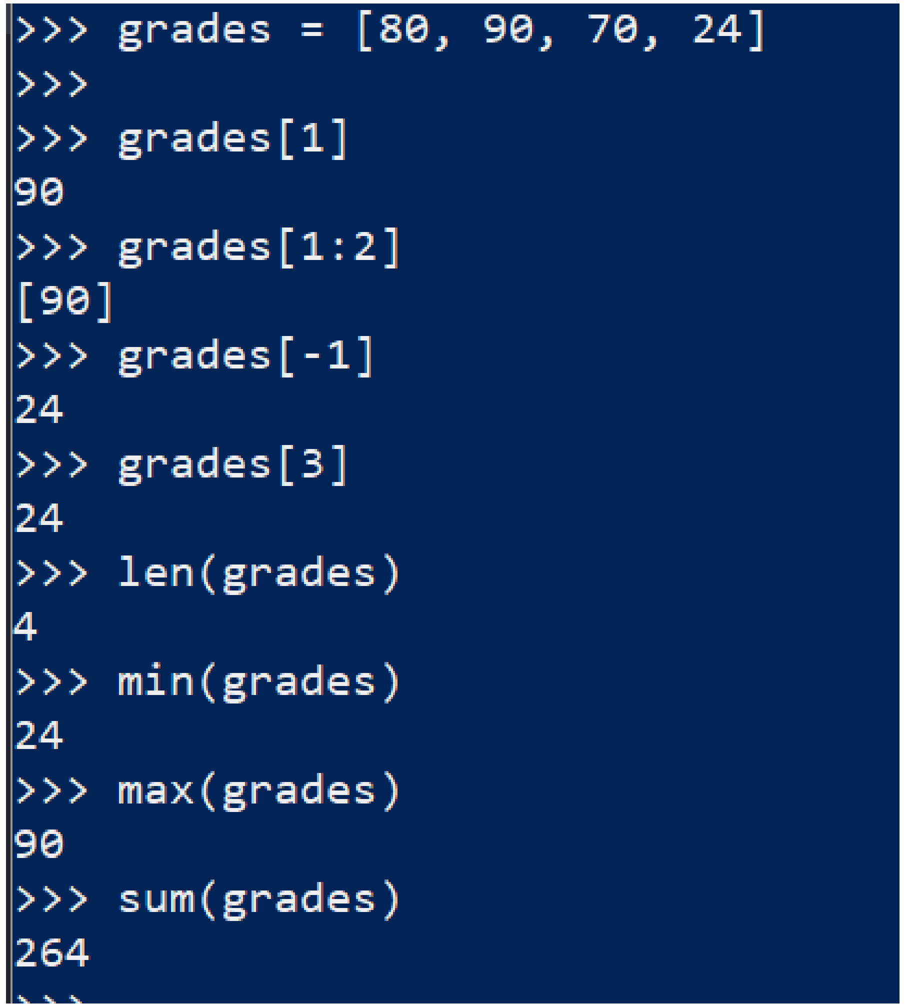
```
names? Alice
How old are you?  d
Oops! your age input was no valid number. Try again...
How old are you?  25
How tall are you in metres?  uj
Oops! your height input was no valid number. Try again...
How tall are you in metres?  1a
Oops! your height input was no valid number. Try again...
How tall are you in metres?  1.78
How much do you weight in kilograms? 7x
Oops! your weight input was no valid number. Try again...
How much do you weight in kilograms? 70
Alice is 25 old, the 1.78 tall and 70.0 heave.
Your BMI is 22.093170054286073
```

Operations on lists

Quiz 6


If the variable grades refers to [80, 90, 70, 24], what does each of the following evaluate to?

grades[1]
grades[1:2]
grade[-1]
grade[3:]
len(grades)
min(grades)
max(grades)
sum(grades)

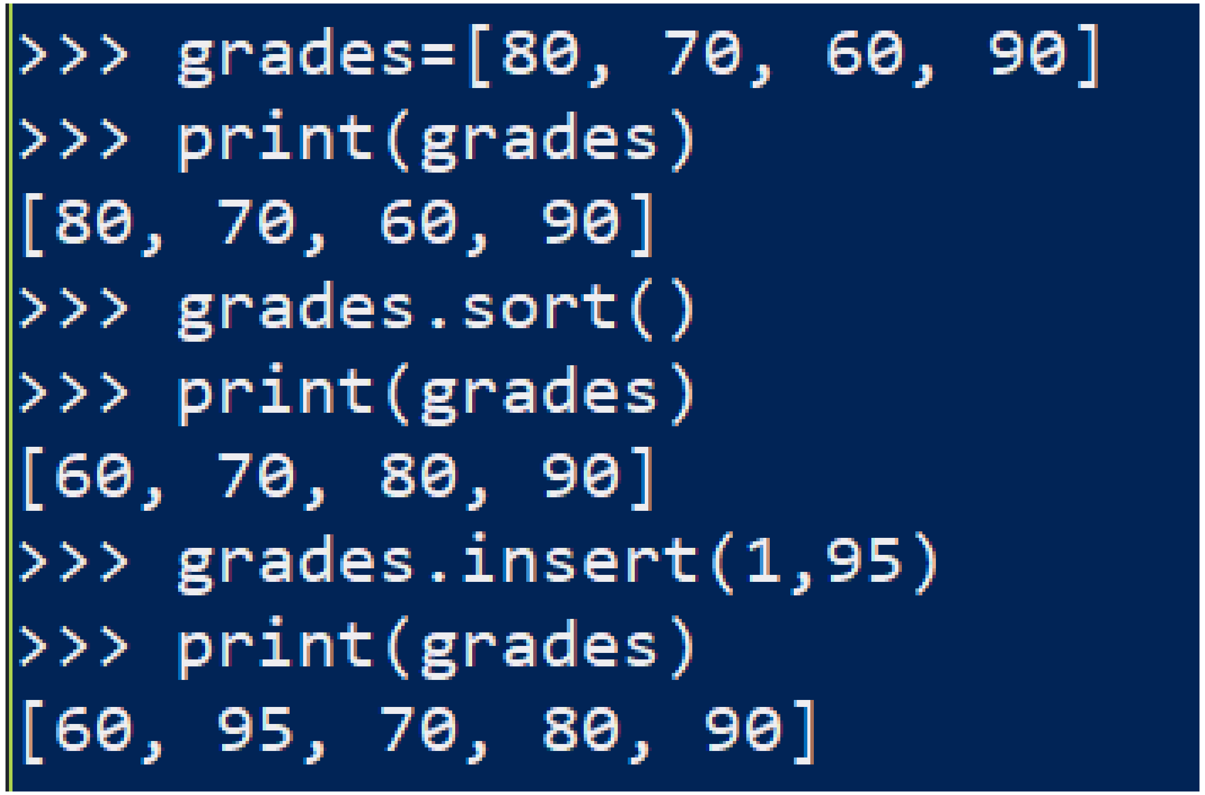

```
>>> grades = [80, 90, 70, 24]
>>>
>>> grades[1]
90
>>> grades[1:2]
[90]
>>> grades[-1]
24
>>> grades[3]
24
>>> len(grades)
4
>>> min(grades)
24
>>> max(grades)
90
>>> sum(grades)
264
```

List[inclusive, exclusive]

Quiz 7

What will grades refer to after this code is executed?

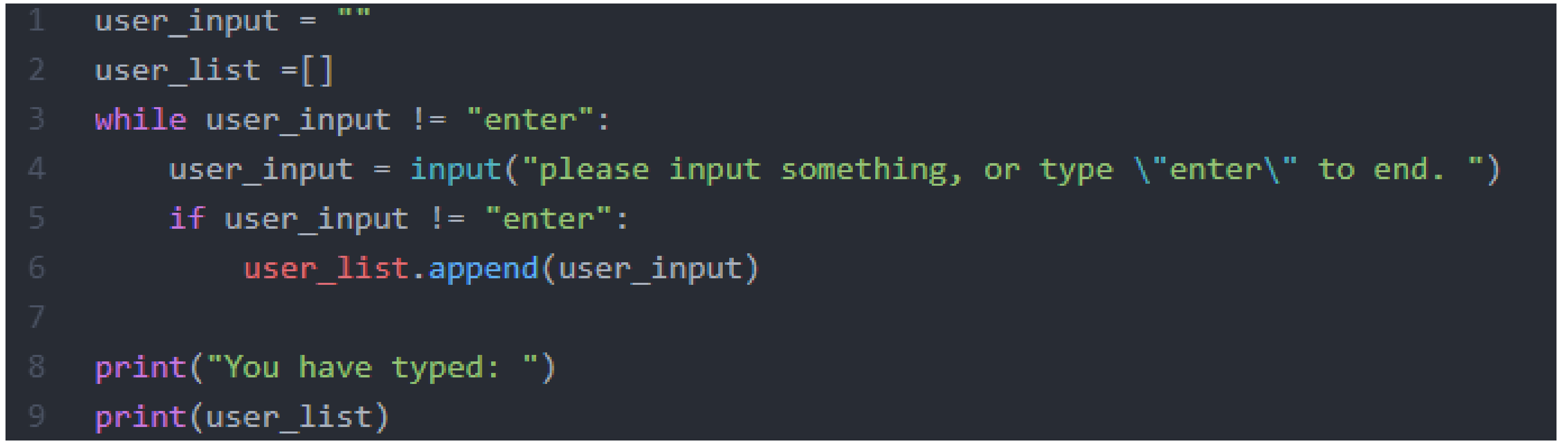
```
grades = [80, 70, 60, 90]
grades.sort()
grades.insert(1, 95)
```



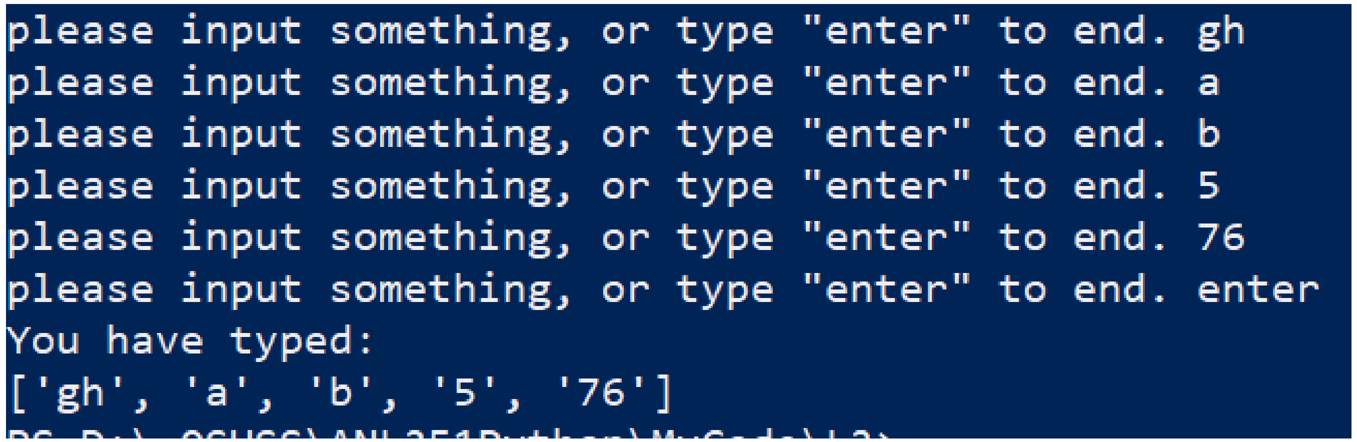
```
>>> grades=[80, 70, 60, 90]
>>> print(grades)
[80, 70, 60, 90]
>>> grades.sort()
>>> print(grades)
[60, 70, 80, 90]
>>> grades.insert(1,95)
>>> print(grades)
[60, 95, 70, 80, 90]
```

Discussion 9

Write a program to add each user input into a list until the user types enter to end.



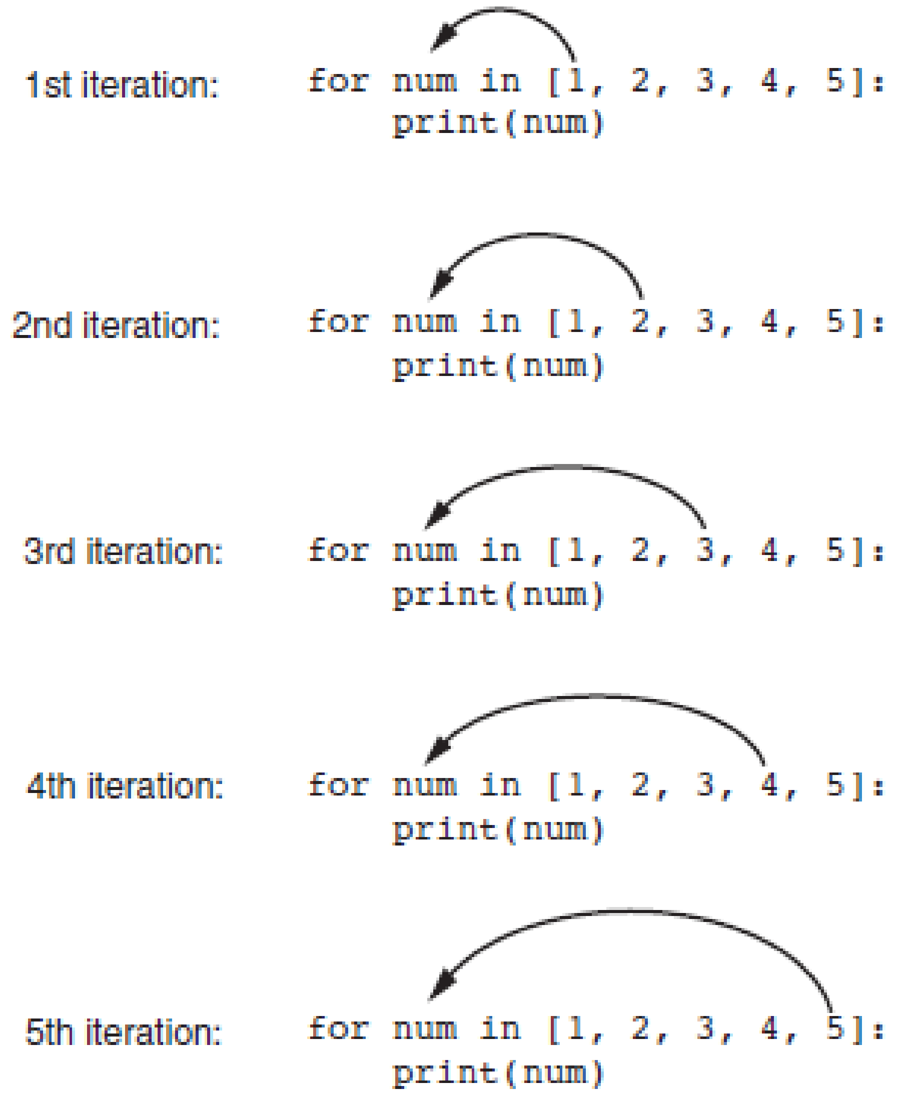
```python
user_input = ""
user_list =[]
while user_input != "enter":
    user_input = input("please input something, or type \"enter\" to end. ")
    if user_input != "enter":
        user_list.append(user_input)

print("You have typed: ")
print(user_list)
```



```
please input something, or type "enter" to end. gh
please input something, or type "enter" to end. a
please input something, or type "enter" to end. b
please input something, or type "enter" to end. 5
please input something, or type "enter" to end. 76
please input something, or type "enter" to end. enter
You have typed:
['gh', 'a', 'b', '5', '76']
```

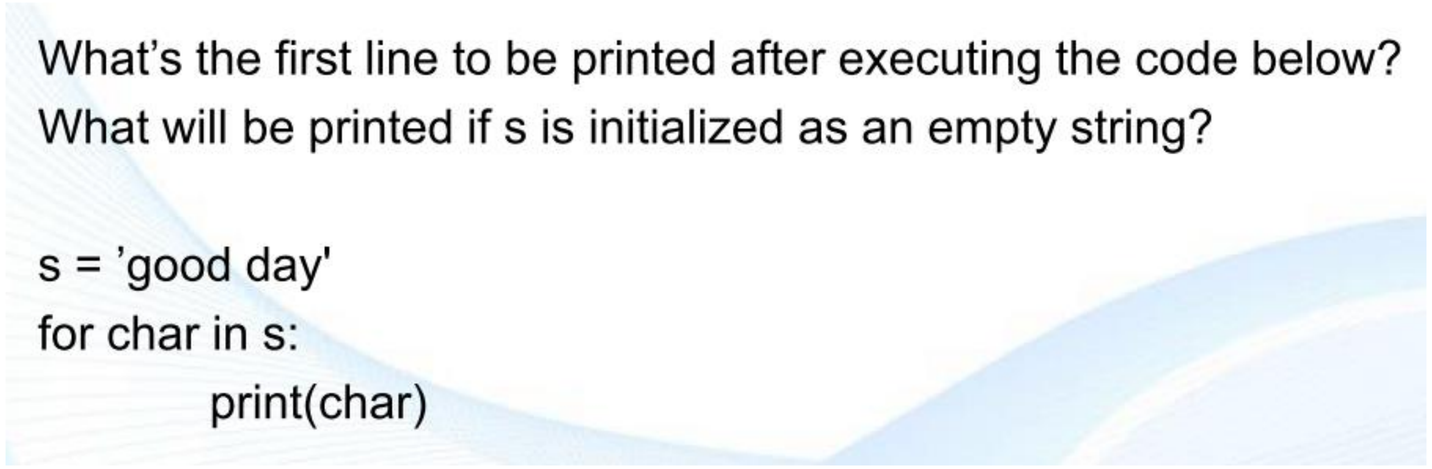
4. Lists and for-loop

Figure 4-4 The for loop


1st iteration:
for num in [1, 2, 3, 4, 5]:
    print(num)
2nd iteration:
for num in [1, 2, 3, 4, 5]:
    print(num)
3rd iteration:
for num in [1, 2, 3, 4, 5]:
    print(num)
4th iteration:
for num in [1, 2, 3, 4, 5]:
    print(num)
5th iteration:
for num in [1, 2, 3, 4, 5]:
    print(num)

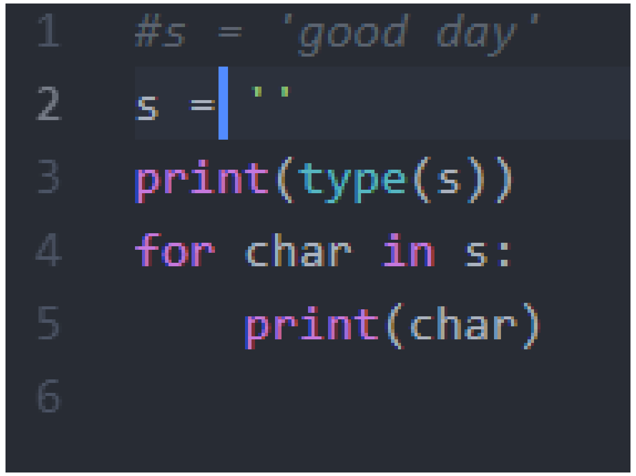
(Toby, 2021)

Discussion 10

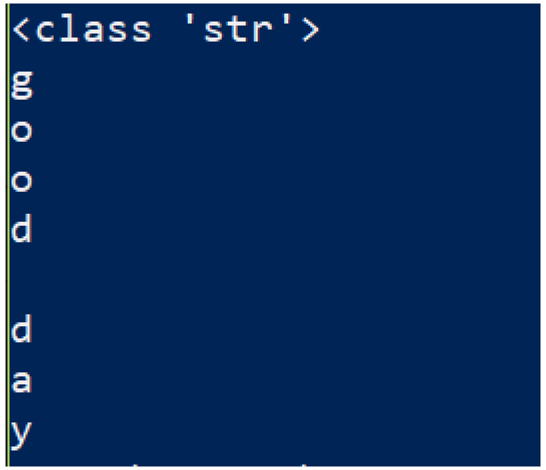


What's the first line to be printed after executing the code below?
What will be printed if s is initialized as an empty string?

```
s = 'good day'
for char in s:
        print(char)
```

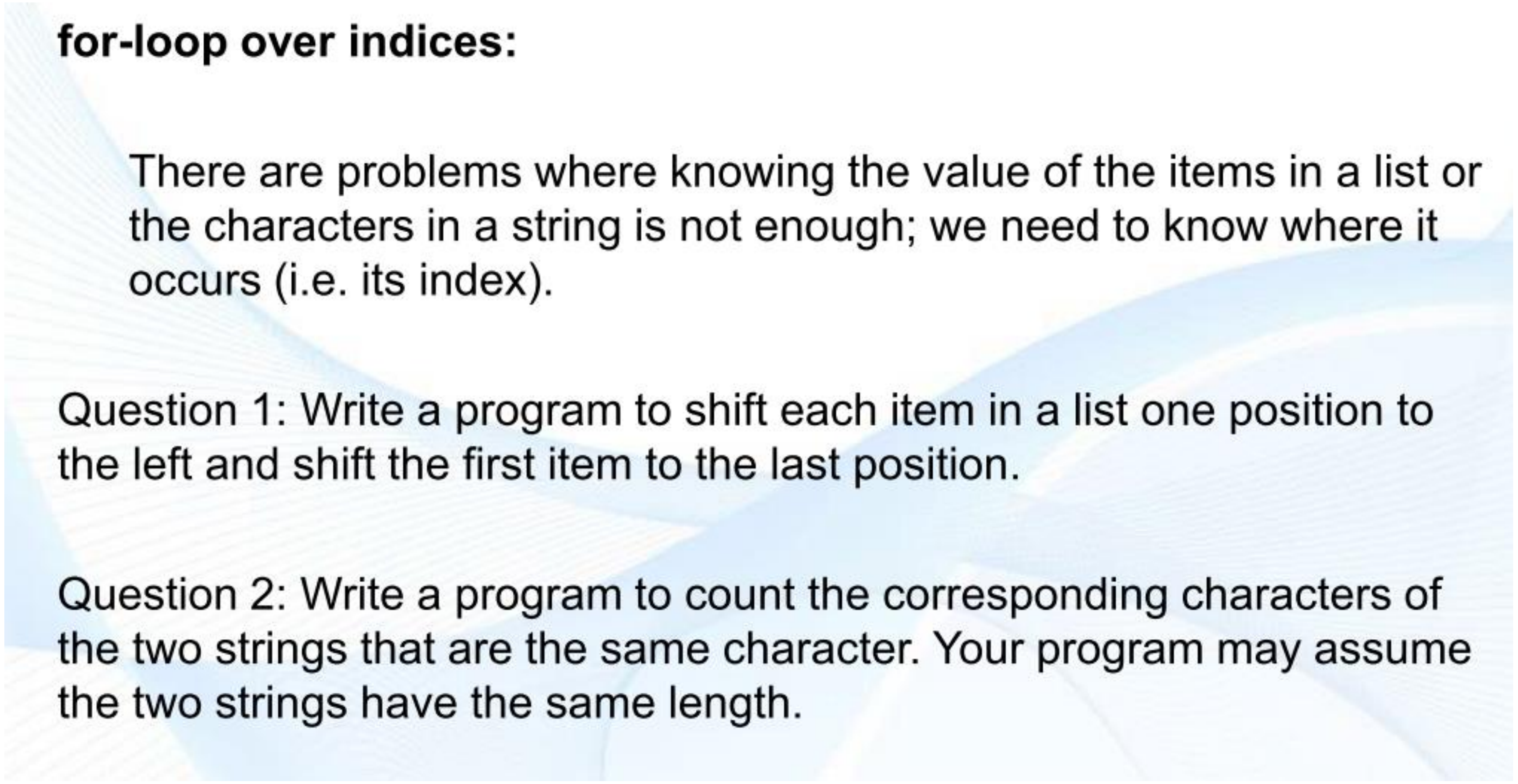


```
#s = 'good day'
s = ''
print(type(s))
for char in s:
    print(char)

```



```
<class 'str'>
g
o
o
d

d
a
y
```

Discussion 11

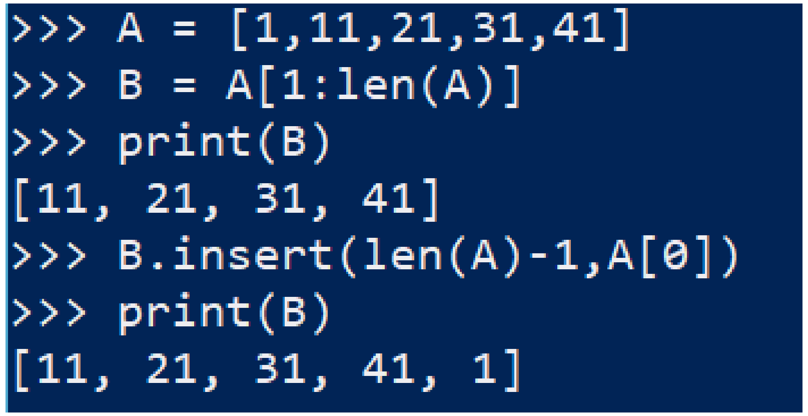


for-loop over indices:

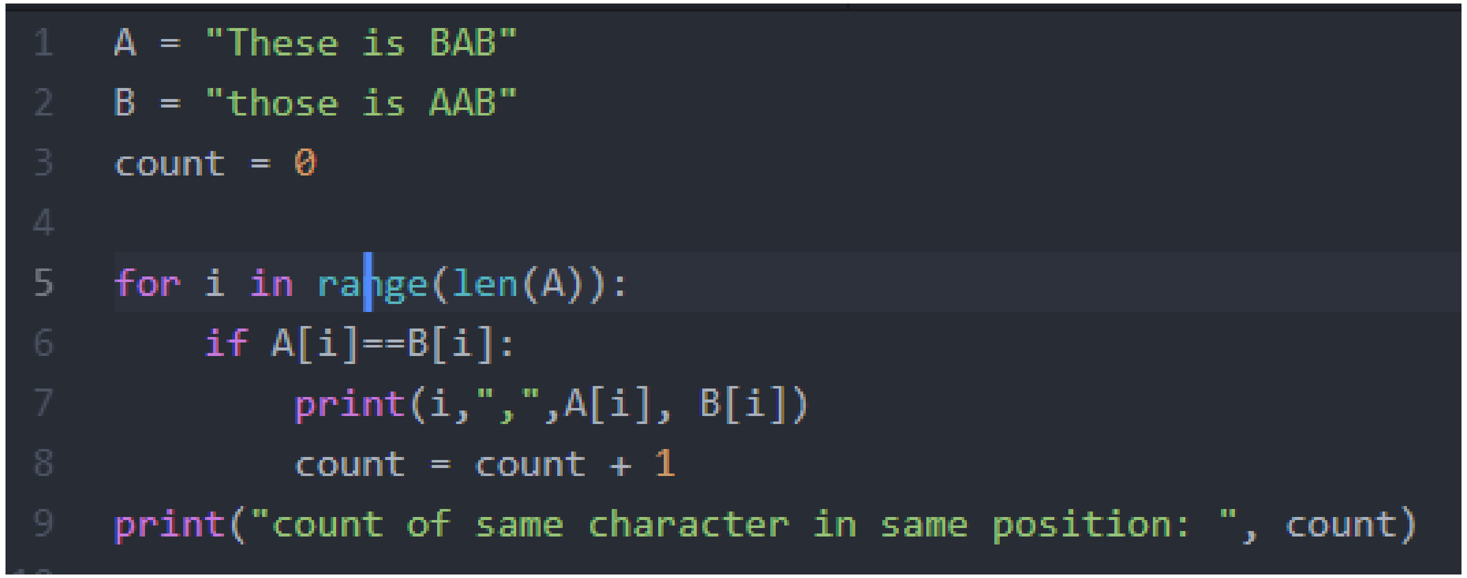
There are problems where knowing the value of the items in a list or the characters in a string is not enough; we need to know where it occurs (i.e. its index).

Question 1: Write a program to shift each item in a list one position to the left and shift the first item to the last position.

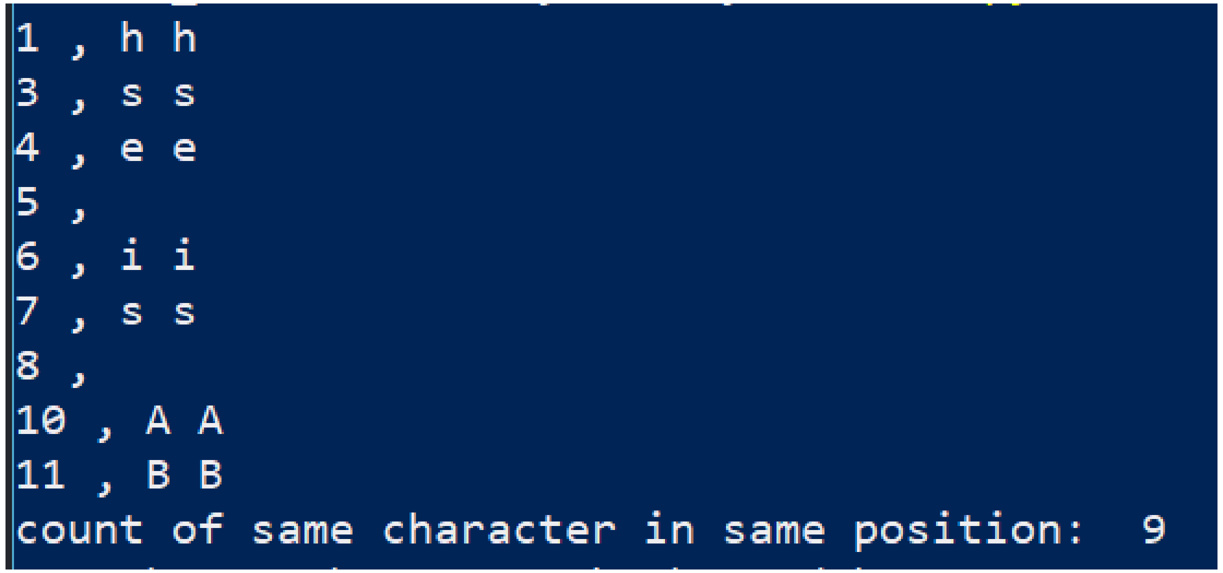
Question 2: Write a program to count the corresponding characters of the two strings that are the same character. Your program may assume the two strings have the same length.



```
>>> A = [1,11,21,31,41]
>>> B = A[1:len(A)]
>>> print(B)
[11, 21, 31, 41]
>>> B.insert(len(A)-1,A[0])
>>> print(B)
[11, 21, 31, 41, 1]
```



```python
A = "These is BAB"
B = "those is AAB"
count = 0

for i in range(len(A)):
    if A[i]==B[i]:
        print(i,",",A[i], B[i])
        count = count + 1
print("count of same character in same position: ", count)
```



```
1 , h h
3 , s s
4 , e e
5 ,
6 , i i
7 , s s
8 ,
10 , A A
11 , B B
count of same character in same position:  9
```

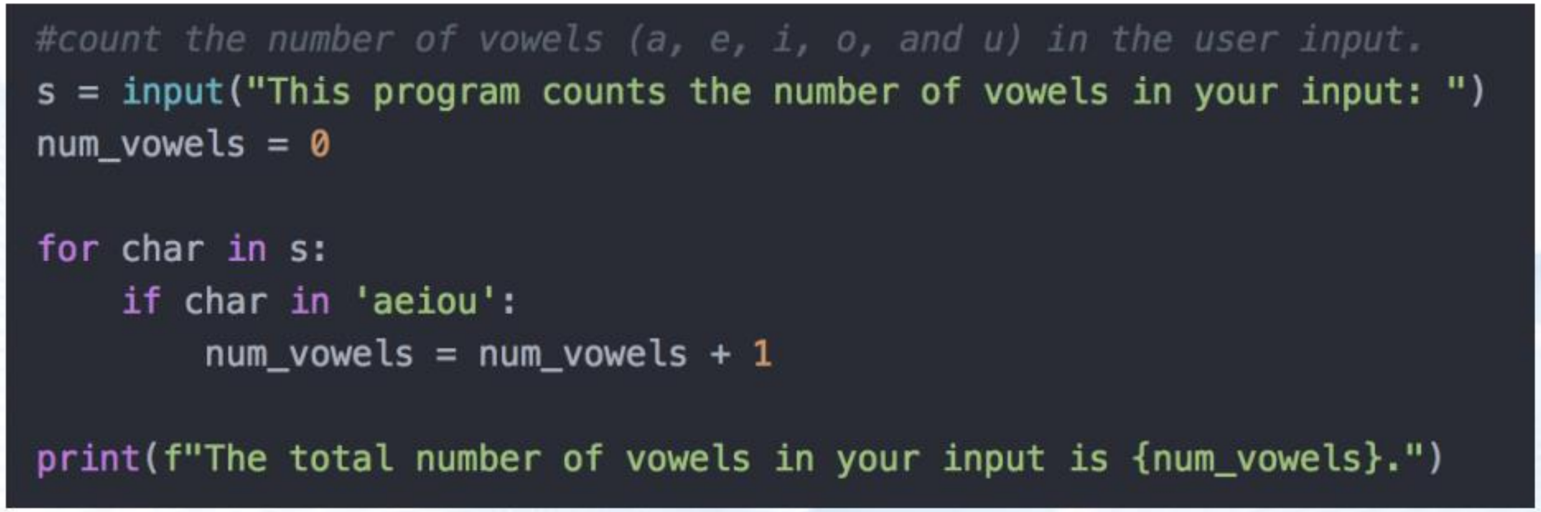
Using the `range` Function with the `for` Loop

- The `range` function simplifies the process of writing a `for` loop
  - `range` returns an iterable object
    - Iterable: contains a sequence of values that can be iterated over
- `range` characteristics:
  - One argument: used as ending limit
  - Two arguments: starting value and ending limit
  - Three arguments: third argument is step value

(Toby, 2021)

Choose test cases

Discussion 12

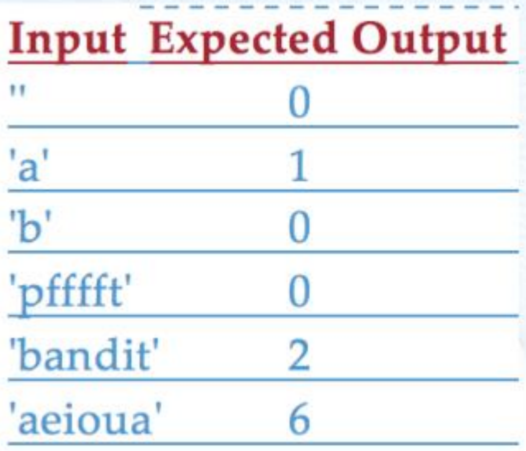


```
#count the number of vowels (a, e, i, o, and u) in the user input.
s = input("This program counts the number of vowels in your input: ")
num_vowels = 0

for char in s:
    if char in 'aeiou':
        num_vowels = num_vowels + 1

print(f"The total number of vowels in your input is {num_vowels}.")
```

Which test cases should you choose in order to thoroughly test your code?

It is not realistic to test using every single possible input. Instead, we create relevant categories, and choose one representative **from each category**.

Discussion: Anymore test cases to add? Can the code above pass all the test cases you choose?



Input	Expected Output
''	0
'a'	1
'b'	0
'pffft'	0
'bandit'	2
'aeioua'	6

Try more normal cases, special cases, extreme cases, and unexpected cases.

References

- Toby Donaldson, Starting out with Python, 2021 (2014)
- Zed A. Shaw, Learn Python 3 **the Hard Way**: A Very Simple Introduction to the Terrifyingly Beautiful World of Computers and Code, July 2017
  - https://learnpythonthehardway.org/python3/
  - http://www.informit.com/promotions/book-registration-learn-python-3-the-hard-way-141409 (videos)

Q&A
Thank you