Fundamentals of Programming 1

Lecture 5

Aurelia Power, ITB, 2018

How to address the issue of multiple if statements – solution 2

- **if** statements are always evaluated, even when the condition has been satisfied.
- For instance, assume I got 73 marks in my FOP 1 exam; all the ranges are still going to be checked, even though we only need one statement to be executed:

```
if (grade >= 80 and grade <= 100): ## will be evaluated
  print("it's  an A") ## won't be executed
if (grade >= 60 and grade < 80): ## will be evaluated
  print("it's  a B") ## will be executed
if(grade >= 40 and grade < 60)  : ## will be evaluated
   print("it's  a C") ## won't be executed
if (grade >= 35 and grade < 40 ):## will be evaluated
  print ("it's  a D") ## won't be executed
if (grade > =0  and grade < 35 ):## will be evaluated
  print("it's  an F") ## won't be executed
```

Multiple alternatives with multiple if-elif…-else

- We can use the **if-elif…-else** construct to re-write the previous code:

```
if (grade >= 80 and grade <= 100): ## will be evaluated
  print("it's  an A") ## won't be executed
elif (grade >= 60 and grade < 80): ## will be evaluated
  print("it's  a B") ## will be executed

elif(grade >= 40 and grade < 60): ## won't be evaluated
   print("it's  a C") ## won't be executed
elif (grade >= 35 and grade < 40 ): ## won't be evaluated
  print ("it's  a D") ## won't be executed
else: ## won't be evaluated
 print("it's  an F") ## won't be executed
```

Multiple alternatives with multiple if-elif…-else

- In **if-elif…-else** constructs the first **if** condition is always evaluated and if it happens to be True the rest will not be evaluated!!
- But if it happens to be False, only then it moves on to check the next condition in the **elif** header and so on…
- So, in our example, the last 4 clauses are no longer evaluated because the program found that the condition in the first elif header is satisfied.
- To summarize, first level ifs are always going to be evaluated.
- Subsequent first level elifs and elses are going to be evaluated only if the previous conditionals are false.

Multiple alternatives – multiple selection

- The general syntax for if-elif…-else construct:

```
if (condition):
   ## conditional instruction(s)/statement(s) go here
elif ( another_condition):
   ## conditional instruction(s)/statement(s) go here
.
.
.
else:
    ## conditional instruction(s)/statement(s) go here
```

Multiple alternatives – multiple selection

- When the program has multiple alternatives/multiple courses of action that the program can take, it is called **multiple selection**;
- Each alternative/course of action is determined/controlled by different Boolean expressions.
- Only **one branch/alternative/one set of instructions will be executed**, depending on which condition/Boolean expression evaluates to true first.
- In multiple selection we can have **as many elif clauses as needed**.
- The **last else clause** can be omitted, depending on the program.

Let's put it all together in a program and run few times…

File Edit Format Run Options Window Help

```
## ask user to input the mark
grade = input("Enter your mark: ")

## convert it to an integer value
grade =  int(grade)

## check the mark against the conditions specified
if (grade >= 80 and grade <= 100):
    print("it's  an A")
elif (grade >= 60 and grade < 80):
    print("it's  a B")
elif (grade >= 40 and grade < 60):
    print("it's  a C")
elif (grade >= 35 and grade < 40 ):
    print ("it's  a D")
else:
    print("it's  an F")
```

```
 RESTART: C:/Users/Aurelia
py
Enter your mark: 23
it's  an F
>>>
 RESTART: C:/Users/Aurelia
py
Enter your mark: 77
it's  a B
>>>
 RESTART: C:/Users/Aurelia
py
Enter your mark: 95
it's  an A
```

Your turn…

1. Write a Python program in which the user enters either 'A', 'B', or 'C'. If 'A' is entered, the program should display the word 'Apple is my first favourite'; if 'B' is entered, it displays 'Berries are my second favourite'; and if 'C' is entered, it displays 'Cheries are my third favourite'. If the user enters something else, the program should display ‘Not a favourite fruit’.

2. Write a program that keeps the scores of a game between 2 players. The program should take input for both scores (as whole numbers) and store them in 2 variables: score1 and score2, and, assuming that the player with the larger score wins, it should output one of the following “player 1 wins”, “player 2 wins”, or “it’s a tie”.

Nested ifs, elifs and else statements

- Sometimes, when we need to be more specific about the conditions and corresponding action(s), we can nest another if inside the block of code that belongs to an if or elif or an else.
- For example, I got 34 points in my FOP exam, and I want the program to tell me whether I passed and what's my grade:
  - So whether it's an A or a B, or a C, or a D the program should print that I passed, and if it's an F should print that I failed
  - One approach is to have a print statement with pass or fail in addition to the print statement for the actual grade

Nested ifs, elifs and else statements

```
if (grade >= 80 and grade <= 100):
        print('you passed')
        print("it's  an A")
elif (grade >= 60 and grade < 80):
        print('you passed')
        print("it's  a B")
elif(grade >= 40 and grade < 60):
        print('you passed')
        print("it's  a C")
elif (grade >= 35 and grade < 40 ):
        print ("it's  a D")
        print('you passed')
else:
        print('you failed')
        print("it's  an F")
```

- But this is a bit tedious, writing the same statement for each applicable range
- We can use another approach…

Nested ifs, elifs and else statements

- The other approach allows us to write a more general condition and nest several sub-conditions:

```
if (grade >= 35 and grade <= 100):
        print('you passed')
        if (grade >= 80):
                print("it's  an A")
        elif (grade >= 60 and grade < 80):
                print("it's  a B")
        elif(grade >= 40 and grade < 60):
                print("it's  a C")
        else:
                print ("it's  a D")
else:
        print('you failed')
        print("it's  an F")
```

Nested ifs, elifs and else statements

- Nested clauses allow us to be more specific;
- We can nest as many ifs or el-ifs as needed;
- Each nested clause can contain further nested ifs… but it is not a good idea to have so many levels of nesting;
- The else statement itself can contain nested ifs and/or elifs and/or else;
- Each nested block has to be further indented;
- Nested ifs are evaluated only if the condition for the enclosing block resolves to True;
- Nested elifs and elses are evaluated only if the condition for the enclosing block resolves to True AND if the previous (same level) if/elif resolves to False.

Let's put it all together and run it…

File Edit Format Run Options Window Help

```
## ask user to input the mark
grade = input("Enter your mark: ")

## convert it to an integer value
grade =  int(grade)

## check the mark against the conditions specified
if (grade >= 35 and grade <= 100):
    print('you passed')
    if (grade >= 80):
        print("it's  an A")
    elif (grade >= 60 and grade < 80):
        print("it's  a B")
    elif (grade >= 40 and grade < 60):
        print("it's  a C")
    else:
        print ("it's  a D")
else:
    print('you failed')
    print("it's  an F")
```

```
 RESTART: C:\Users\Aure
py
Enter your mark: 23
you failed
it's  an F
>>>
 RESTART: C:\Users\Aure
py
Enter your mark: 97
you passed
it's  an A
>>>
 RESTART: C:\Users\Aure
py
Enter your mark: 37
you passed
it's  a D
>>>
 RESTART: C:\Users\Aure
py
Enter your mark: 77
you passed
it's  a B
>>>
```

Nested ifs, elifs and else statements

```
if (conditionA):
   ## conditional instruction(s)/statement(s) go here
   if (conditionA1):
      ## conditional instruction(s)/statement(s) go here
   elif(conditionA2):
      if(conditionA21):
          ## conditional instruction(s)/statement(s) go here
   else:
       ## conditional instruction(s)/statement(s) go here
.
.
.
else:
   if(conditionB1):
      ## conditional instruction(s)/statement(s) go here
```

Nested ifs, elifs and else statements

Exercises:

1. Modify the previous program to also account for ranges that are negative and print appropriate statements.
2. Modify the program further by using equivalent expressions
3. What does the following block of code output?

```
r = 10; x = 0; n = 'hello'
if ( r >= 10):
    print('r is  greater or equal than 10')
    if (x < 5 and len(n) == 3):
        print('x > 5')
    elif( x >= 5 and len(n) >3):
        print('x >= 5')
    else:
        print('something...')
else:
    print('goodbye')
```

Ternary Operator

- Ternary operator takes 3 operands
- It is used to describe if-else conditional statements in a shorter way:
- The general syntax:

<expression1> if <condition> else <expression2>

- expression1 is evaluated if condition is true, otherwise expression2 is evaluated.

- Example:

```
print('underage' if age <18 else 'adult')

## if the value of age is less than 18 it will print underage, if the value of age is greater or equal to 18 it will print adult

EXERCISE: what is the output of the following?

print('pass' if grade >= 35 else 'fail')

print('well' if 'well' not in 'wellness' else 'hmmm...')
```

Ternary Operator

- Ternary expressions can be quite complex embedding several ternary expressions:

print('child' if age < 13 else 'teenager' if age < 18 else 'adult')

if the value of age is less than 18 and less than 13 it will print child, if the value of age is less but greater or equal to 13 it will print teenager, and if the value of age is greater or equal to 18 it will print adult

- What will the following code print for the values of hour of 7, 14 and 21?

print('morning' if hour < 12 else 'afternoon' if hour < 19 else 'evening')

- What will the following code print for the values of hour of 7, 11, 14, 20 and 21?

print('ungodly hour' if hour < 10 else 'morning' if hour < 13 else 'afternoon' if hour < 19 else 'evening' if hour < 21 else 'night')

Abbreviated Assignment Expressions

Python provides several assignment operators for abbreviating assignment expressions that uses the same variable on both sides of the assignment, using the following **format**:

variable operator= expression

Example 1: c = c + 3

Can be abbreviated with the addition & assignment operator **+=** as: c **+=** 3

Example 2: c = c - 3

Can be abbreviated with the subtraction & assignment operator **-=** as: c **-=** 3

Example 3: c = c * 3

Can be abbreviated with the multiplication & assignment operator ***=** as: c ***=** 3

Abbreviated Assignment Expressions

Operator	Shorthand Expression	Actual Expression	Explanation
+=	a += b	a = a + b	It first adds a and b and then re-assigns that value to a
-=	a -= b	a = a - b	It first minuses a and b and then re-assigns that value to a
*=	a *= b	a = a * b	It first multiplies a and b and then re-assigns that value to a
/=	a /= b	a = a / b	It first divides a by b and then re-assigns that value to a
%=	a %= b	a = a % b	It first divides a and b and then re-assigns the remainder of that division to a
**=	a **= b	a = a ** b	It first raises a at the power of b and then re-assigns that value to a
//=	a //= b	a = a // b	It first floor divides a by b and then re-assigns that value to a

Ternary Operator and Abbreviated Assignments

- Exercises:

1. What is the value of each variable after executing the following code?

```
x = 10; y = 4; z = 'E'
x %= y; y *= z; z  = x // 3; z /= x
```

2. Write a program that will prompt the user to enter 2 integer values, apply all the abbreviated assignment operators and output the values of variables after each assignment.

3. Write a program that outputs the alpha grade based on the numerical grade: > 80 is A, between 60 and 79 is a B, between 40 and 59 is a C, between 35 and 39 is a D, and less than 35 is an F.

RECAP

- Python Language
- Variables
- Basic output
- Basic Input
- Basic float and string formatting
- Mathematical Operators
- String concatenation and operator overloading
- Character representation
- Relational Operators

RECAP

- Membership Operators
- Logical Operators
- Operator Precedence
- Computational Problem Solving
- Errors
- Sequence Structure
- Selection Structure: single, double and multiple
- Ternary conditionals
- Abbreviated Assignments