Booleans and Conditional Statements

Beste Filiz Yuksel

Variables and their Names

Must start with letter (a-z, A-Z) or “_”

fooBar, foo_bar, _fooBar

Cannot contain space: foo bar

Cannot start with number: 37fooBar

Cannot have special characters:

foo-bar, fooBar$

Cannot be a Python keyword, e.g., return

Boolean Variables and Expressions

"Boolean" means, it's *True* or *False*

```
print(True)
type(True)
type(False)
```

Boolean Variables and Expressions

Boolean variables can *only* be True or False (has to be capitalized first letter)

```
>>> canVote = True
>>> canVote
True
>>>type(canVote)
<class 'bool'>
```

Boolean Variables and Expressions

Boolean variables can *only* be True or False

```
>>> myScore = 100
>>> yourScore = 99
>>> iWon = myScore > yourScore
>>> iWon
True
```

Boolean Relational Operations

Greater than: >

Less than: <

Greater than or equal to: >=

Less than or equal to: <=

Equal to: == (***two*** equal signs!)

Not equal to: !=

Boolean Expressions

```
>>> 3 > 4
False
>>> 7.555 < 7.556
True
>>> 1.0000001 >= 1
True
>>>3==3
True
>>>3!=3
False
```

String Comparisons

A String is a string or sequence of characters inside quotation marks “ ” or ‘ ’.

Remember computers do not store characters such as A, B, C, but numeric codes that represent characters.

A-Z represented by numbers 65-90.

a-z represented by numbers 97-122.

When a program compares characters, it is comparing the numeric codes for the characters.

```
>>> 'a'<'b'
True
```

String Comparisons

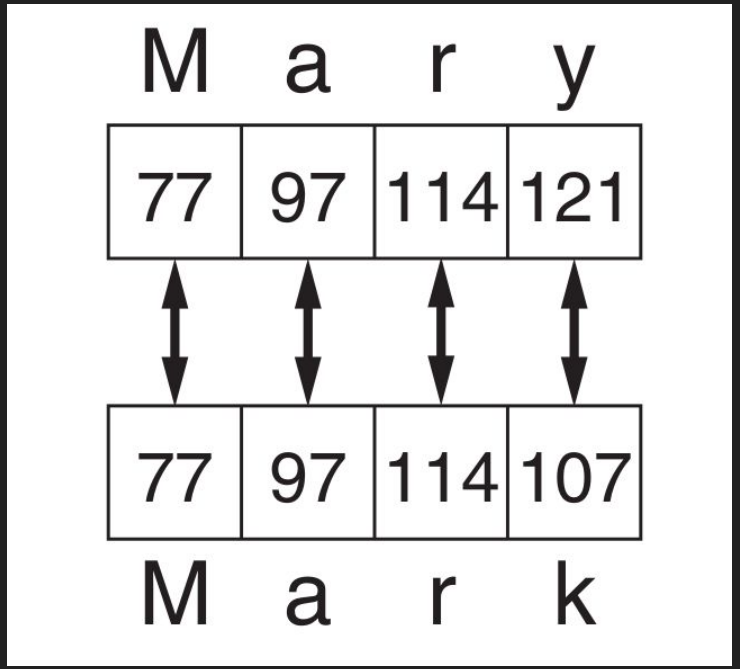
Would this be True or False?

```
>>> 'Mary'<'Mark'
```

String Comparisons

Would this be True or False?

```
>>> 'Mary'<'Mark'
```



‘M’ in Mary compared with 'M’ in Mark (same, next character is compared)

‘a’ in Mary compared with ‘a’ in Mark (same, next character is compared)

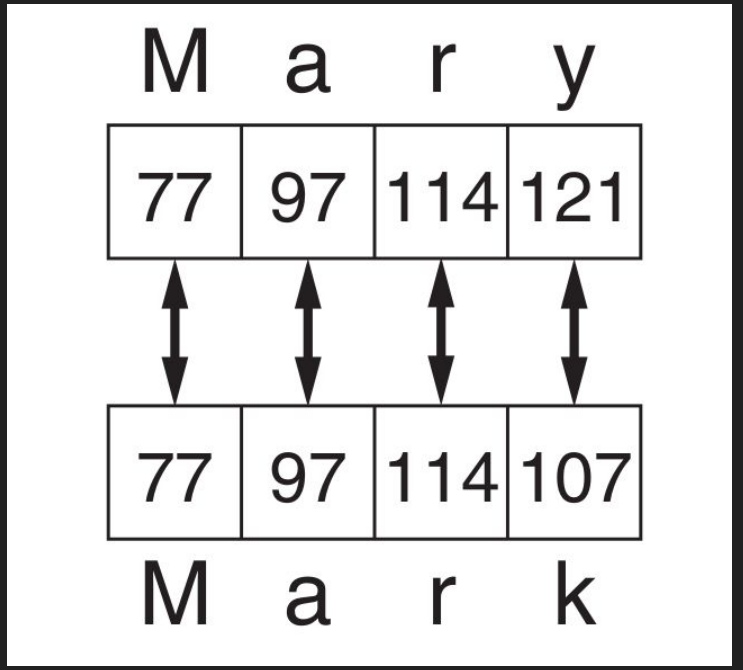
‘r’ in Mary compared with ‘r’ in Mark (same, next character is compared)

‘y’ in Mary compared with ‘k’ in Mark. The character ‘y’ has higher ASCII code (121) than ‘k’ (107).

String Comparisons

Would this be True or False?

```
>>> 'Mary'<'Mark'
False
```



‘M’ in Mary compared with 'M’ in Mark (same, next character is compared)

‘a’ in Mary compared with ‘a’ in Mark (same, next character is compared)

‘r’ in Mary compared with ‘r’ in Mark (same, next character is compared)

‘y’ in Mary compared with ‘k’ in Mark. The character ‘y’ has higher ASCII code (121) than ‘k’ (107).

Control Structures

A *control structure* is a logical design that controls the order in which a set of statements execute.

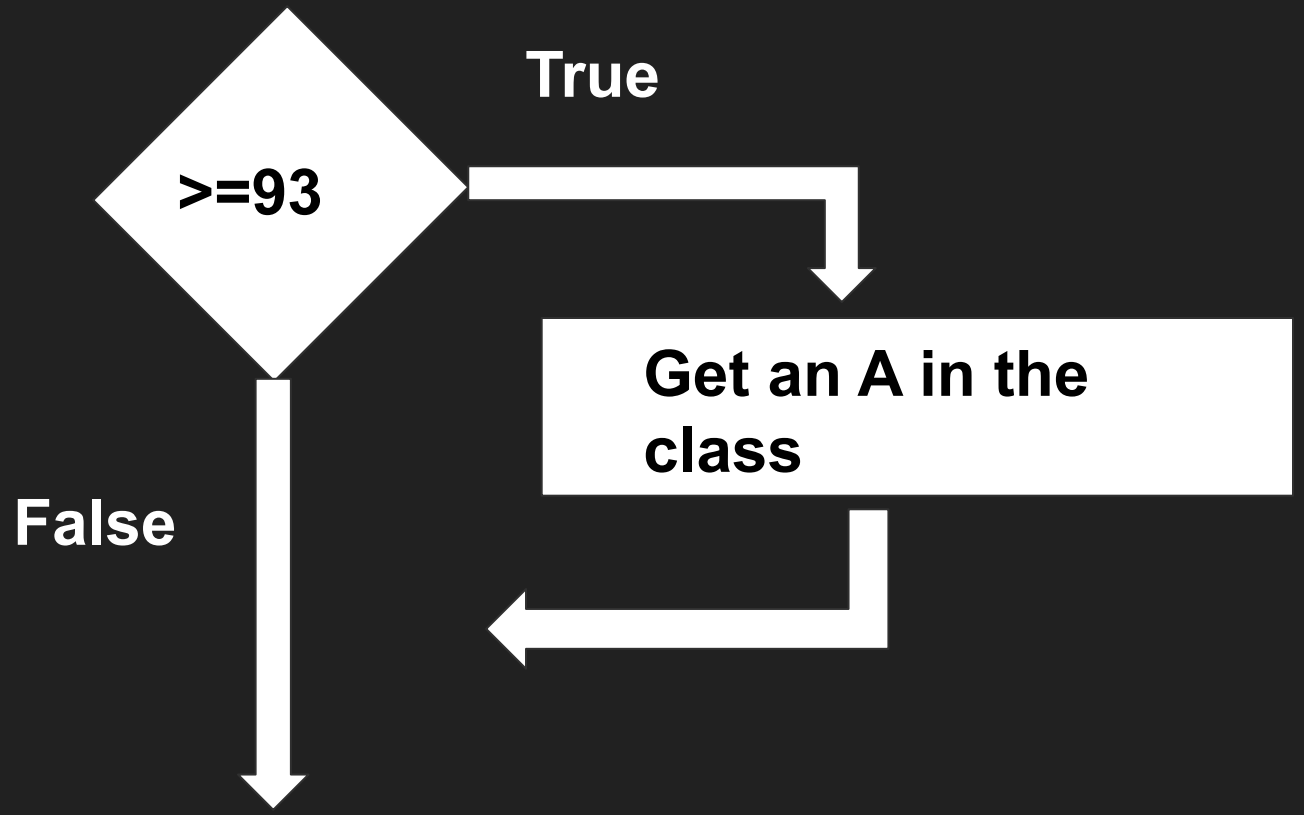
So far we have used a simple control structure called *sequence structure* where each statement is executed in order. E.g.

```
name = input('What is your name?')
age = int(input('What is your age?'))
print('Name: ', name)
print('Age: ', age)
```

Decision Structures

But sometimes you would only want certain actions performed depending on certain conditions.

For example, if you get 93% or higher in this class you will get an A.

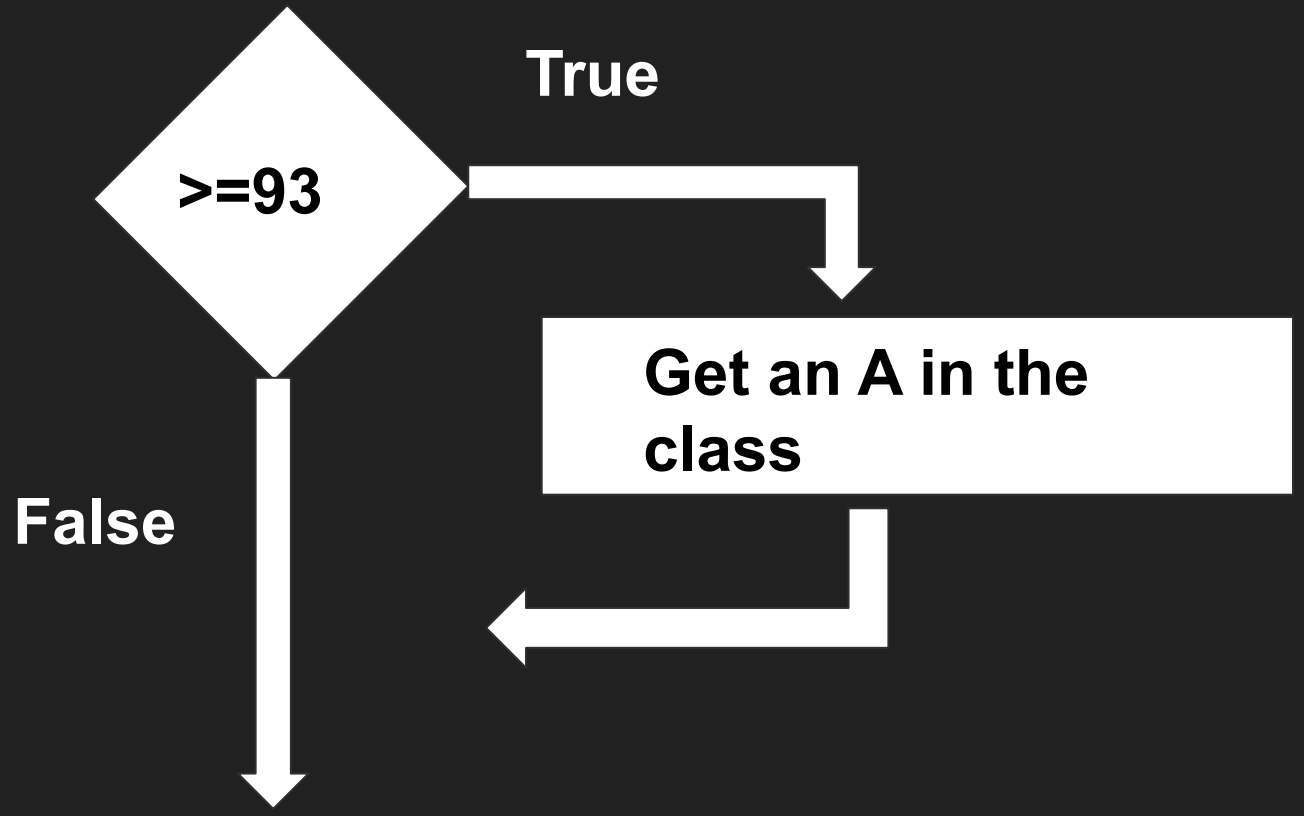


Decision Structures

In this example diamond shape represents some condition that must be tested.

If this condition is true, the student receives an A in the class.

If the condition is false the action is skipped.



The `if` statement

The `if` statement is used for a single alternative decision structure. Here is the general format

if *condition*:

    statement

    statement

    statement

    etc.

Indentation is required because Python interpreter uses it to understand where the block begins and ends.

The `if` statement

```
grade=93
if grade>= 93:
   print(‘You got an A in class!’)
```

if <boolean expression>:

    <code that executes only if above expression is true>

Try writing this code in a script and running it.

Try it with and without the indentation and see what happens.

Try it with different values of `grade`.

The `if` statement

```
x = 1
y = 2
if x>y:
   print(‘x is greater than y’)

if x<y:
   print(‘x is less than y’)

if x==y:
   print(‘x is equal to y’)
```


Bill
created Microsoft

The `if-else` Statement

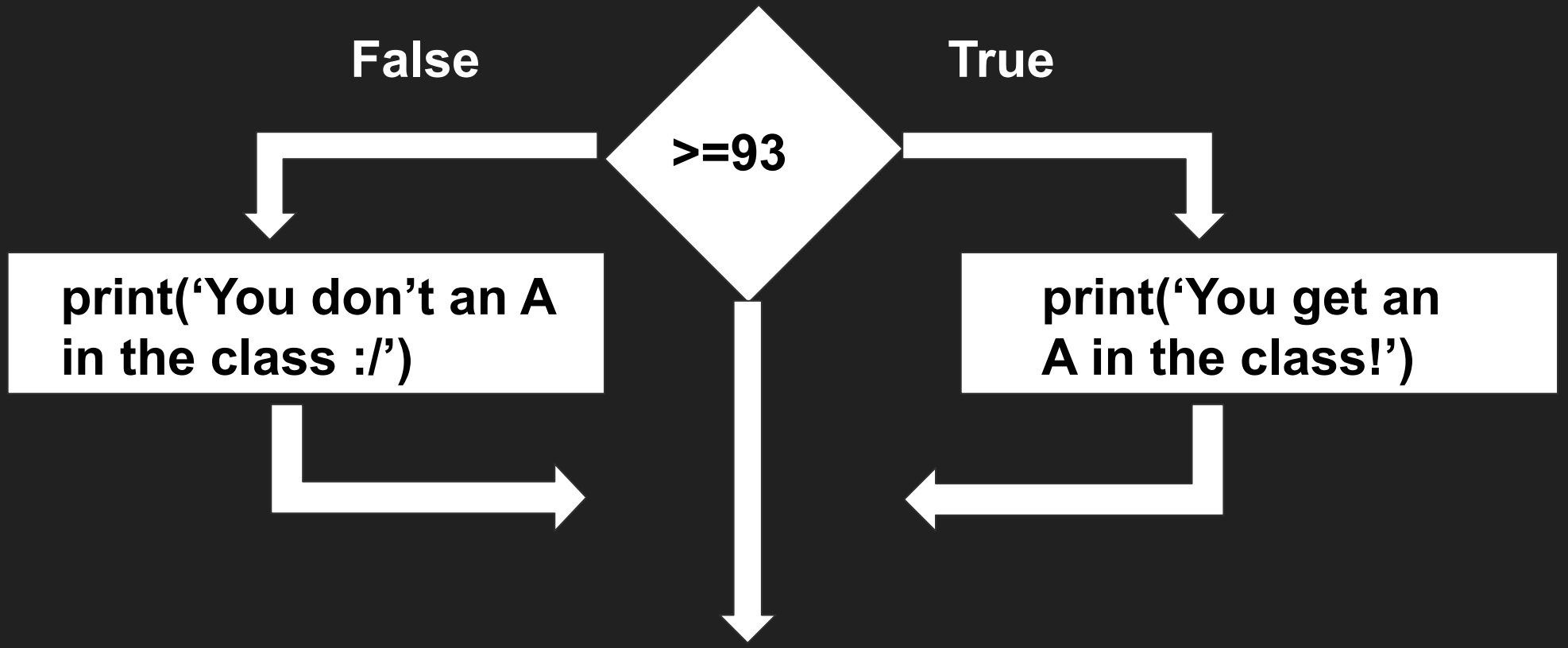
The if-else statement will execute one block of statements if its condition is true, or another block if its condition is false.

The if statement was a *single* alternative decision structure.

The if-else statement is a dual alternative decision structure. It has *two* possible paths of execution.

One path is taken if a condition is true, the other path is taken if the condition is false.

The `if-else` Statement



The `if-else` statement

The `if-else` statement is used for a dual alternative decision structure. Here is the general format

if *condition*:
    statement
    statement
    etc
else:
    statement
    statement
    etc

The `if-else` statement

```
if grade>= 93:
   print(‘You got an A in class!’)
else:
   print(‘You didn’t get an A in class :/’)
```

if <boolean expression>:

    <code that executes only if above expression is true>

else:

    <code that executes only if above expression is false>

The `if-else` statement

```
x = 1
y = 2
if x>y:
   print(‘x is greater than y’)
else:
   print(‘x is less than or equal to y’)
```

We could have used a third clause for x==y, we will cover that soon!

`else` is *optional* second part of `if` statement

Let’s take a break and do questions 1-3 from In-Class Exercise 3

Nested Conditionals and if-elif-else Statement

Review `if` statement

```
if grade>= 93:
   print(‘You got an A in class!’)
```

if <boolean expression>:

    <code that executes only if above expression is true>

Review `if-else` statement

```
if grade>= 93:
   print(‘You got an A in class!’)
else:
   print(‘You didn’t get an A in class :/’)
```

if <boolean expression>:

    <code that executes only if above expression is true>

else:

    <code that executes only if above expression is false>

Nested `if` statements

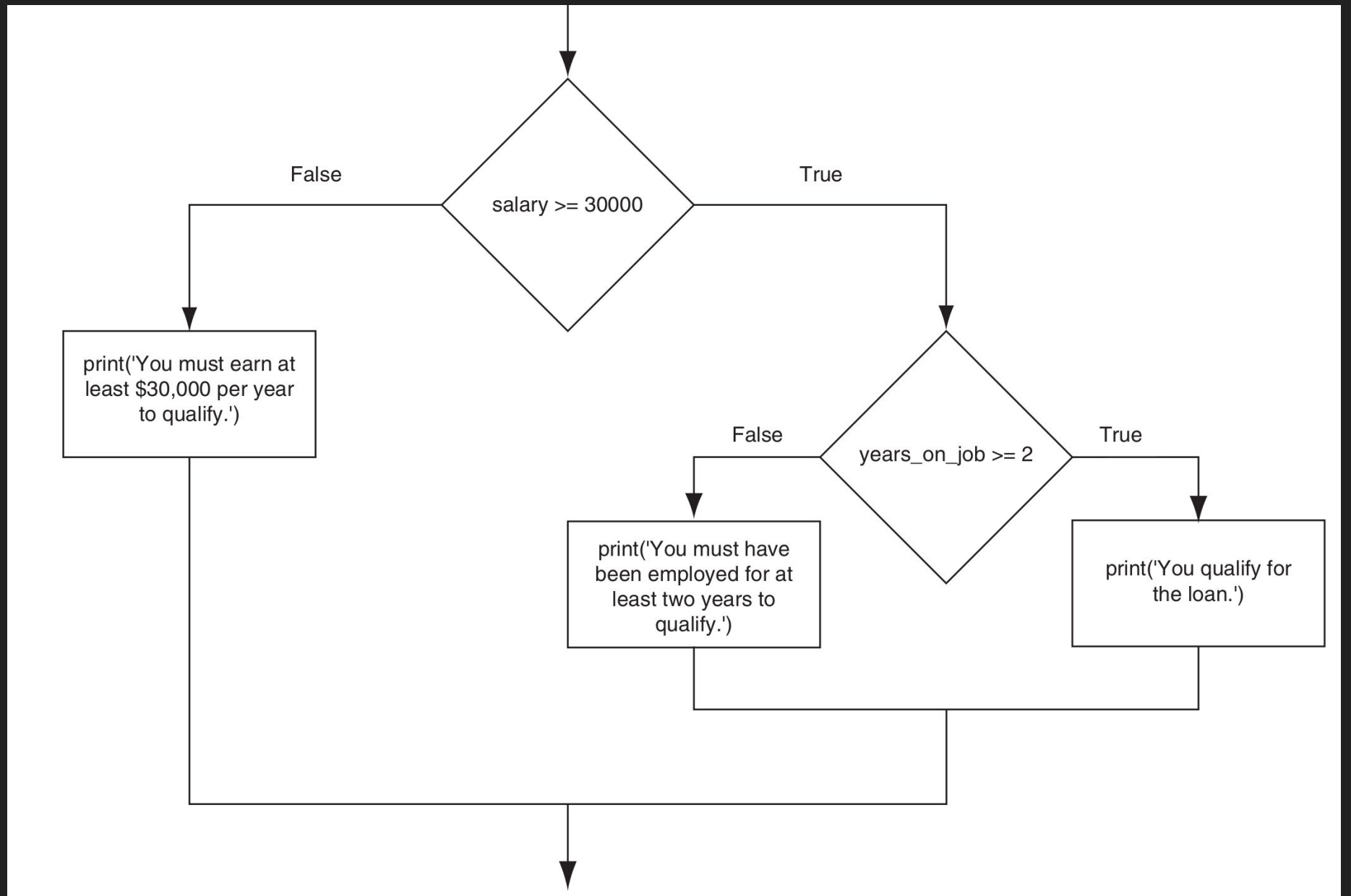
You can nest decision structures inside other decision structures.

Example:

A Python program that determines whether a bank customer qualifies for a loan.

To qualify two conditions must be met:

1. Customer must earn $30,000 per year, and
2. Customer must have been employed for at least two years.


salary >= 30000
False
True
print('You must earn at least $30,000 per year to qualify.')
years_on_job >= 2
False
True
print('You must have been employed for at least two years to qualify.')
print('You qualify for the loan.')

```
MIN_SALARY = 30000.0
MIN_YEARS = 2
salary = float(input('Enter your annual salary: ')
years_on_job = int(input('Enter the number of years employed: ')

if salary >= MIN_SALARY:
    if years_on_job >=MIN_YEARS:
        print('You qualify for the loan.')
    else:
        print('You must have been employed for at least two years.')
else:
    print('You must earn at least $', format (MIN_SALARY, ',.2f'), ' per year to
     qualify.', sep= ' ')
```

Nesting `if` Statements under `if`

```
if <boolean expression>:
    statement 1
    if <boolean expression>:
        statement 2
        statement 3
    statement 4
else:
    statement 5
statement 6
```

Nesting `if` Statements under `else`

```
if <boolean expression>:
    statement 1
    statement 2
    …
else:
    if <boolean expression>:
        statement 3
        statement 4
    statement 5
statement 6
```

Example: Nested `if` Statements

```
x = 42
y = 42
if x < y:
    print("x is less than y")
else:
    if x > y:
        print("x is greater than y")
    else:
        print("x and y must be equal")
```

The `if-elif-else` Statement

The if-elif-else statement allows for a simpler type of logic.

```
if condition_1:                 # if condition_1 is true the block of statements under it are
    statement                     executed and the rest of the structure is ignored
    statement
elif condition_2:               # if condition_2 is true the block of statements under it are
    statement                     executed and the rest of the structure is ignored
    statement
else:                           # if all previous conditions are false then these statements
    statement                     are executed
    statement
```

Conditionals using `elif`

```
x = 10
y = 10
if x < y:
    print("x is less than y")
elif x > y:
    print("x is greater than y")
else:
    print("x and y must be equal")
```

Let’s compare nested conditionals with elif

1. Ask user to enter a mark for class.
2. If the score is greater than or equal to 93, then the grade is A.

    Else, if the score is greater than or equal to 83, then the grade is B.

        Else, if the score is greater than or equal to 73, then the grade is C.

            Else, if the score is greater than or equal to 63, the the grade is D.

                Else, the grade is F.

Nested Conditionals Example

```
score = int(input('Enter your test score: '))

if score >= 93:
    print ('I got an A, I love Python')
else:
    if score >= 83:
        print ('Your grade is B')
    else:
        if score >=73:
            print ('Your grade is C')
        else:
            if score >= 63:
                print ('Your grade is D, please go to office hours.')
            else:
                print('Your grade is F :/')
```

```
score = int(input(‘Enter your test score: ’))

if score >= 93:
    print (‘I got an A, I love Python’)
elif score >= 83:
    print (‘Your grade is B’)
elif score >=73:
    print (‘Your grade is C’)
elif score >= 63:
    print (‘Your grade is D, please go to office hours.’)
else:
    print(‘Your grade is F :/’)
```

elif example

order matters - what happends if we change the order?

```
score = int(input(‘Enter your test score: ’))

if score >= 73:
        print (‘Your grade is C’)
elif score >= 83:
        print (‘Your grade is B’)
elif score >=93:
        print (‘I got an A, I love Python’)
elif score >= 63:
        print (‘Your grade is D, please go to office hours.’)
else:
        print(‘Your grade is F :/’)
```

What if the order was like this? What grade would a student who received a score of 99 get?