Lecture 8 – Conditional Execution

Conditional statements are part of every programming language.

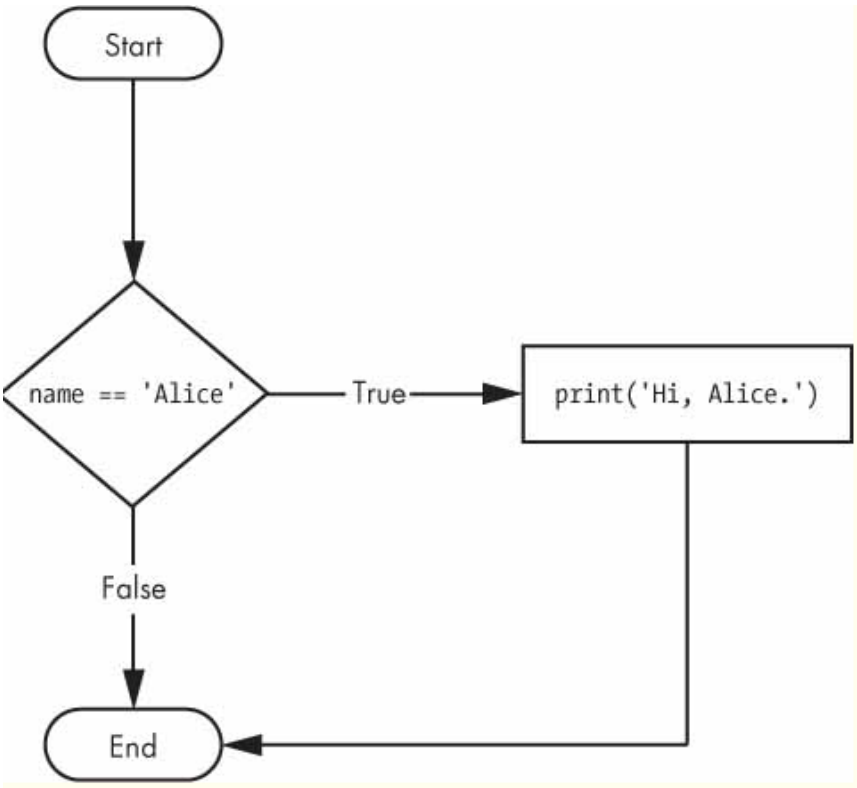
With conditional statements, we can have code that sometimes runs and at other times does not run, depending on the conditions of the program at that time.

When we fully execute each statement of a program, moving from the top to the bottom with each line executed in order, we are not asking the program to evaluate specific conditions.

By using conditional statements, programs can determine whether certain conditions are being met and then be told what to do next.

If statement

The if statement will evaluate whether a statement is true or false, and run code only in the case that the statement is true.



For example

```
grade = 70          #  Variable grade with value 7
if grade >= 65:     print("Passing grade")
```

The **variable grade** is giving the integer value of 70.

The if statement is used to **evaluate** whether or not the variable grade is greater than or equal (>=) to 65.

If it is , the **print** out the string Passing grade.

Output is → Passing grade

What is the output if the variable grade has a value of 60 ?

Another Example

```
balance = -5
if balance < 0:     print("Balance is below 0, add funds now or you will be charged a penalty.")
```

Output is :

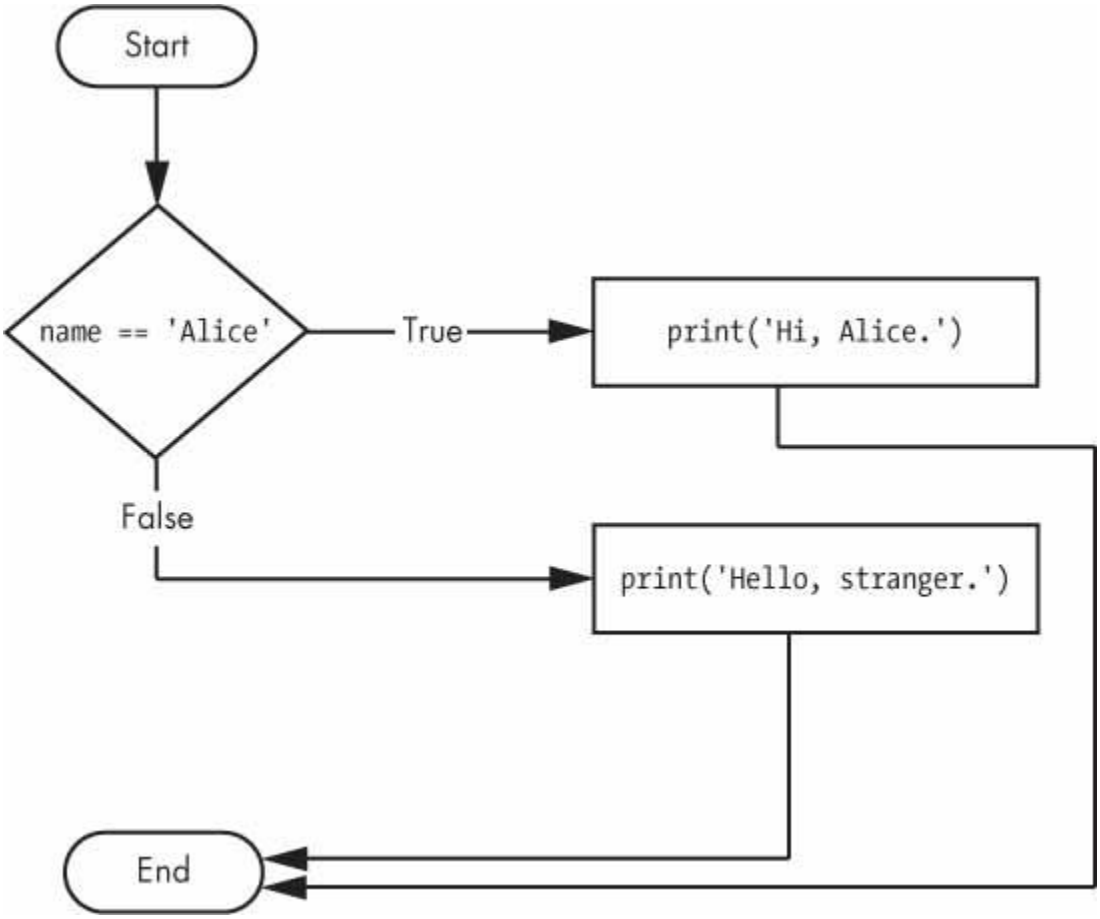
Balance is below 0, add funds now or you will be charged a penalty.

What is the output of the code segment?

```
grade = 70
if grade < 60:    print("\n Failing")
if grade < 70:    print("\n Average")
if grade < 80:    print("\n Good")
if grade < 90:    print("\n Very Good")
if grade <= 100:    print("\n Excellent ")
```

Else Statement

If you would like the program to do something even when an if statement evaluates to false then use the else statement



Example

```
grade = 60
if grade >= 65:    print("Passing grade")
else:    print("Failing grade")
```

Output

```
Failing grade
>>>
```

Else if statement

If you would like for the program to evaluates more than two possible outcomes.

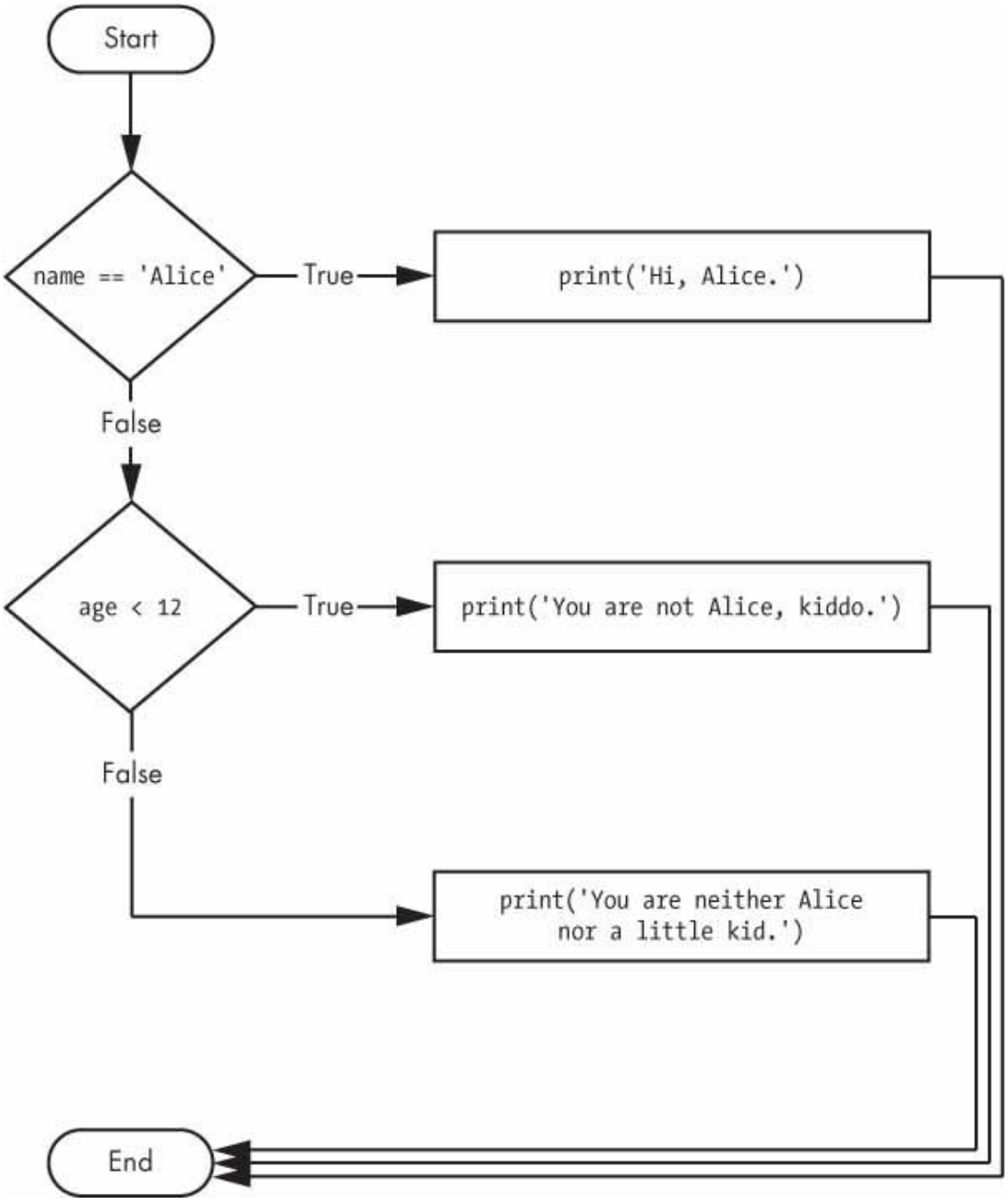
Use an else if statement, which is written in Python as elif.

The elif or else if statement looks like the if statement and will evaluate another condition.

In the bank account program, one may have three discrete outputs for three different situations:

- **The balance is below 0**
- **The balance is equal to 0**
- **The balance is above 0**

The elif statement will be placed between the if statement and the else statement.


Start
name == 'Alice'
True
print('Hi, Alice.')
False
age < 12
True
print('You are not Alice, kiddo.')
False
print('You are neither Alice
nor a little kid.')
End

Example

```
balance = 50
if balance < 0:
   print("Balance is below 0, add funds now or you will be charged a penalty.")
elif balance == 0:
   print("Balance is equal to 0, add funds soon.")
else:    print("Your balance is above  0.")
```

Sample Run

```
Your balance is 0 or above.
```

Q: What is the output if the balance is 0 or -5

Example

In the **grade.py** program, there are a few letter grades corresponding to ranges of numerical grades:

90 or above is equivalent to an A grade
80-89 is equivalent to a B grade
70-79 is equivalent to a C grade
65-69 is equivalent to a D grade
64 or below is equivalent to an F grade

Write a code to implement the above requirement

Solution

```
grade    = int( input("Enter Your Grade  ") )

if grade    >= 90:   print("A grade")
elif grade >=80:    print("B grade")
elif grade >=70:    print("C grade")
elif grade >= 65:   print("D grade")
else:    print("Failing grade")
```

Sample Run

```
Enter Your Grade  70
C grade
>>>
```

Q: What if the user enters -10 or A ???

Q: What happens if you reverse the logical expressions such as using <= 90 instead of >= 90 and so on and entering values such 95 , 65 , or -10 ???

Example

```
grade    = int( input("Enter Your Grade  ") )

if grade    <= 90:   print("A grade")
elif grade <=80:    print("B grade")
elif grade <=70:    print("C grade")
elif grade <= 65:   print("D grade")
else:    print("Failing grade")
```

Sample Run

```
Enter Your Grade  95
Failing grade
>>>

Enter Your Grade  65
A grade
>>>

Enter Your Grade  -10
A grade
>>>
```

IF statement and range of integers

Example

```
my_int = int(input("Enter a number between 1 and 50 --- >   "))

list = range(0 , 50)

result = my_int in list
if result == 1:
     print( my_int , " is in the list\n")
else:
     print( my_int , " is not in the list\n")
```

Sample run

```
Enter a number between 1 and 50 --- >   40
40  is in the list


>>>
Enter a number between 1 and 50 --- >   49
49  is in the list

>>>

Enter a number between 1 and 50 --- >   50
50  is not in the list

>>>

Enter a number between 1 and 50 --- >   0
0  is in the list

>>>
```

```
Enter a number between 1 and 50 --- >   A
Traceback (most recent call last):
  File "C:\Users\HP\AppData\Local\Programs\Python\Python38-
32\Program1.py", line 1, in <module>
    my_int = int(input("Enter a number between 1 and 50 --- >   "))
ValueError: invalid literal for int() with base 10: 'A'
```

IF statement and range of integers

Example

```
my_int = int(input("Enter a number between 1 and 50 --- >   "))

list = range(0 , 50 , 2)    # Only Even Numbers

result = my_int in list
if result == 1:
     print( my_int , " is in the list\n")
else:
     print( my_int , " is not in the list\n")
```

Sample Run

```
Enter a number between 1 and 50 --- >   0
0  is in the list

Enter a number between 1 and 50 --- >   50
50  is not in the list

>>>

Enter a number between 1 and 50 --- >   48
48  is in the list

>>>
Enter a number between 1 and 50 --- >   19
19  is not in the list

>>>
```

Nested If Statements

Nested if statements are used for situations if you would like to check for a secondary condition if the first condition executes as true.

Thus , you can have an if-else statement inside of another if-else statement

```
if statement1:                  #outer if statement
        print("true")

    if nested_statement:        #nested if statement
        print("yes")
    else:                       #nested else statement
        print("no")
else:                           #outer else statement
    print("false")
```

A few possible outputs can result from this code:

- **If statement1 evaluates to true, the program will then evaluate whether the nested_statement also evaluates to true. If both cases are true, the output will be:**

```
true
yes
```

- **If, however, statement1 evaluates to true, but nested_statement evaluates to false, then the output will be:**

  true
  no

- **And if statement1 evaluates to false, the nested if-else statement will not run, so the else statement will run alone, and the output will be:**

  false

  One can also have multiple if statements nested throughout our code

```
grade    = int( input("Enter Your Grade  ") )
if grade >= 65:
    print("Passing grade of:")
    if grade >= 90:         print("A")
    elif grade >=80:        print("B")
    elif grade >=70:        print("C")
    elif grade >= 65:       print("D")
else:    print("Failing grade")
```

Example :

A program to determine whether a character that is entered from the keyboard is numeric, alphanumeric, small letter, capital letter , or a digit

Solution

```
s=input("Enter any character:  ")
if s.isalnum():
   print("Alpha Numeric Character")
   if s.isalpha():
      print("Alphabet character")
      if s.islower():
         print("Lower case alphabet character")
      else:
         print("Upper case alphabet character")
   else:
     print("it is a digit")
elif s.isspace():
   print("It is space character")
```

Sample Run

Enter any character: **A**

Alpha Numeric Character

Alphabet character

Upper case alphabet character

>>>

Enter any character: **2**

Alpha Numeric Character

it is a digit

>>>

Enter any character:

It is space character

>>>

Enter any character: **a**

Alpha Numeric Character

Alphabet character

Lower case alphabet character

>>>

If & Compound Boolean Expressions

To be eligible to graduate from Loyola University Chicago, you must have 120 credits *and* a GPA of at least 2.0. This translates directly into Python as a ***compound condition***:

```
credits >= 120 and GPA >=2.0
```

This is true if both `credits >= 120` is true and `GPA >= 2.0` is true.

Example:

```
credits = float(input('How many units of credit do you have? '))
GPA = float(input('What is your GPA? '))

if credits >= 120 and GPA >=2.0:
    print('You are eligible to graduate!')
else:
    print('You are not eligible to graduate.')
```

Sample Run

```
How many units of credit do you have? 135
What is your GPA? 1.25
You are not eligible to graduate.
>>>

How many units of credit do you have? 125
What is your GPA? 3.5
You are eligible to graduate!
>>>
```

Example

```
isSunday = True
isMonday = True
isHoliday = False

if isHoliday and isSunday:
    print('\nSunday is a Funday!!')
else:
    print('\nNot holiday !! Please start working :(')


if isHoliday or isSunday:
    print('\nSunday is a Funday!!')
else:
    print('\nNot holiday !! Please start working :(')


isHoliday = True

if isHoliday and isSunday:
    print('\nSunday is a Funday!!')
else:
    print('\nNot holiday !! Please start working :(')


isHoliday = False

if isHoliday and isSunday and isMonday:
    print('\nSunday is a Funday!!')
else:
    print('\nNot holiday !! Please start working :(')


if isHoliday or isSunday or isMonday:
    print('\nSunday is a Funday!!')
else:
    print('\nNot holiday !! Please start working :(')
```

Sample Run

```
        Not holiday !! Please start working :(

        Sunday is a Funday!!

        Sunday is a Funday!!

        Not holiday !! Please start working :(

        Sunday is a Funday!!
>>>
```

Switch Statement?

You'll often need to evaluate expressions against multiple values. Modern programming languages such as C++ offer `switch` statements to do this.

The pseudocode syntax for the **`switch`** statement is as follows:

```
switch expression          # the expression has a certain value

case 1:                    # we test the value
        do something       # the program performs an action depending on the value
 case 2:
        do something
  case 3:
        do something
 default:                  # if none of the case statements is satisfied
         do something      # the program performs a default action
```

In the above code, *expression* gets compared against the values in each *case* clause. Once a matching *case* clause is found, the code inside that *case* clause is run. If there is no matching *case* clause for `expression`, the code under the `default` clause gets executed.

Assume that you're building a program to check a computer's processor. Based on the result, the program will let the gamer know if their processor is compatible with a certain video game. Using if statement , the program is as follows :

```
print("\n\nPlease Select Your CPU model: ")
print ("\n\tCeleron")
print ("\tCore i3")
print ("\tCore i5")
print ("\tCore i7")
print ("\tCore i9")

# First, ask the player about their CPU
cpuModel = str.lower(input("\n\n CPU Model is ---- >>   "))
if cpuModel == "celeron":
   print ("\nForget about it  ...")

elif cpuModel =="core i3":
  print ("\nWeak Processor ;)")

elif cpuModel == "core i5":
  print ("\nGood Processor, you should be fine.")

elif cpuModel == "core i7":
  print ("\nVery Good Processor !")

elif cpuModel == "core i9":
  print ("\nExcellent Processor ...")

else:
  print ("\nUnknown CPU Model ?")
```

Python match-case statement is used to simulate the switch-case statement. This simple and effective feature is available in Python 3.11.1

A match statement will compare a given variable's value to different shapes, also referred to as the pattern. The main idea is to keep on comparing the variable with all the present patterns until it fits into one

The match case consists of three main entities:

- **The match keyword**
- **One or more case clauses**
- **Expression for each case**

Example :

Assume that you're building a program to check a computer's processor. Based on the result, the program will let the gamer know if their processor is compatible with a certain video game. The program is as follows :

```
print("\n\nPlease Select Your CPU model: ")
print ("\n\tCeleron")
print ("\tCore i3")
print ("\tCore i5")
print ("\tCore i7")
print ("\tCore i9")

# First, ask the player about their CPU
cpuModel = str.lower(input("\n\n CPU Model is ---- >>   "))

# The match statement evaluates the variable's value
match cpuModel:
   case "celeron":
      print ("\nForget about it ...")
   case "core i3":
      print ("\nWeak Processor ... )")
   case "core i5":
      print ("\nGood Processor ...")
   case "core i7":
      print ("\nVery Good Processor ...")
   case "core i9":
      print ("\nExcellent Processor ...")
   case _:          # the underscore character is used as a invalid type.
      print ("\nUnknown CPU Model ?")
```

Example :

Matching Multiple Patterns in a Single case.

The code will print 1 if the country is either ‘USA’ or ‘Canada’.

```
country = 'Canada'
match country:
        case 'Brazil':
            print(55)
        case 'USA' | 'Canada':
            print(1)
        case 'Japan':
            print(91)
        case _:
            print('Other Country')
```

Match – Case and if statement

Using match – case statements, write code will accept an integer from the keyboard and determine whether the number is positive, negative, or zero.

Solution :

```
n = int(input("Enter an Integer Number   "))
match n:
    case n if n < 0  :
        print("Number is negative")
    case n if n == 0:
        print("Number is zero")
    case n if n > 0:
        print("Number is positive")
```

Match – Case and Logical Operators

Using match – case statements, write code will accept a character from the keyboard and determine whether the character is small letter, capital letter , digit , or a symbol

Solution

```
ch = input("\n\nPlease Enter Your Own Character : ")

match ch :
    case ch if(ch >= 'a' and ch <= 'z')  :
        print("The Given Character ", ch, "is Lower Case Alphabet")
    case ch if (ch >= 'A' and ch <= 'Z'):
        print("The Given Character ", ch, "is Upper Case  Alphabet")
    case ch if(ch >= '0' and ch <= '9'):
        print("The Given Character ", ch, "is a Digit")
    case _:
        print("\nThe Given Character ", ch, "is a Special Character")
```

Exercises :

1. What is the output of the following code ?

```
num = 3
if num > 0:
    print(num, "is a positive number.")
print("This is always printed.")
```

```
num = -1
if num > 0:
    print(num, "is a positive number.")
print("This is also always printed.")
```

2. Write a Python Program to check character is Alphabet Digit or Special Character with using the string functions such as isalpha , isdigit .. etc

3. Simulate a simple calculator using the nested if statement.

The calculator uses + , - , * , / , % , and power

The user must enter 2 integers and an operator.

4. Simulate a simple calculator using match-case statement.

The calculator uses + , - , * , / , % , and power

The user must enter 2 integers and an operator.