COMPSCI 101

Principles of Programming

Lecture 11 – if ... else, if ... elif statements, nested ifs

Learning outcomes

At the end of this lecture, you will be able to use:

- conditional statements which contain an `else` block (`if…else` statements)
- nested ifs
- `if…elif` statements

Recap

From lecture 10

- boolean expressions evaluate to either `True` or `False`
- There are only two boolean values `True` and `False`
- Relational operators (>, <, <=, <= and ==) are used to compare values
- Logical operators (not, and, or) can be used to build more complex boolean expressions
- an if statements is used when a block of code is to be executed only if a particular condition is True

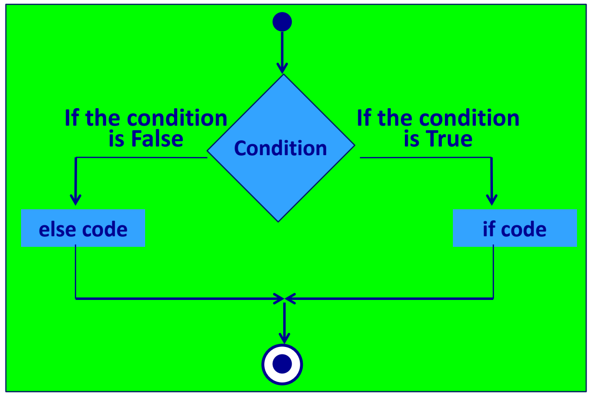
```
def copyright_check(current_y, death_y):
   if current_y - author_death_y >= 50:
       print("Out of copyright")

def main():
   current_year = 2020
   author_death_year = input("Enter year of author's death: ")
   author_death_year = int(author_death_year)
   copyright_check(current_year, author_death_year)

main()
```

```
Enter year of author's death: 1960
Out of copyright
```

```
Enter year of author's death: 1971
```

Python syntax for an if…else statement

In an **`if…else`** statement the code in the 'if block' is executed if the condition evaluates to `True` and the code in the 'else block' is executed if the condition evaluates to `False`.



```
if boolean_expression:
    statement1
    statement2
else:
    statement3
    statement4
```

if…else statement - example

```python
def what_to_wear(temperature):
  if temperature > 25:
      print("Wear shorts.")
  else:
      print("Not hot today!")
      print("Wear long pants.")
  print("Enjoy yourself.")

def main():
  what_to_wear(20)
  print()
  what_to_wear(30)

main()
```

```
Not hot today!
Wear long pants.
Enjoy yourself.

Wear shorts.
Enjoy yourself.
```

Give the output

```python
def show_output(number):
  if number >= 45 and number < 60:
      print("A")
      number = number - 10
  else:
      print("B")
      number = number + 10
  if number % 9 == 0:
      print("C")
      number = number - 5
  else:
      print("D")
      number = number + 6
  print(number)
def main():
  show_output(45)
main()
```

Complete the function

Complete the `add_bonus()` function which prints "Good job!" and returns 30000 plus the salary if the parameter is a value greater than 150000. Otherwise it prints "Superb performance!" and returns 300 plus the salary.

```python
def add_bonus(salary):




def main():
  salary = 34000
  new_salary = add_bonus(salary)
  print("Was: $" + str(salary), "Now: $" + str(new_salary))
  print()
  salary = 250000
  new_salary = add_bonus(salary)
  print("Was: $" + str(salary), "Now: $" + str(new_salary)))
main()
```

```
Superb performance!
Was: $34000 Now: $34300

Good job!
Was: $250000 Now: $280000
```

Nested if's - example

Any statements, including other if statements, can be used inside if statements. For example:

```python
def ice_cream_info(scoops, with_extras, on_cone):
  price = scoops * 1.50
  message = "scoops: " + str(scoops)
  if with_extras:
    message = message + ", plus extras"
    if on_cone:
      message = message + ", on cone"
      price = price + 2
    else:
      message = message + ", in cup"
      price = price + 1
  else:
    if on_cone:
      message = message + ", on cone"
      price = price + 2
    else:
      message = message + ", in cup"
      price = price + 1

  print(message + " $" + str(price))
```

Three calls to the ice_cream_info() function

```python
def main():
  ice_cream_info(3, True, False)
  ice_cream_info(2, False,False)
  ice_cream_info(4, True, True)
main()
```

```
scoops: 3, plus extras, in cup $5.5
scoops: 2, in cup $4.0
scoops: 4, plus extras, on cone$8.0
```

Give the output

```
def display_output(x, y, z):
  if x == 5 or y > 5:
    if x > 4 and z == 8:
      print("A")
    else:
      if y == 6 and z >= x:
        print("B")
      else:
        print("C")
  else:
    print("D")

def main():
  display_output(4, 6, 8)

main()
```

Note how the indentation increases at every nested if and this moves the code further and further to the right hand side.

Executing one of several options

Sometimes you have a situation when you wish to execute one block of code from many options, e.g. if you wish to print one statement depending on the number entered by the user.

```
def what_to_do_now():
  message = "Time to "
  user_choice = int(input("Enter selection (1, 2,
                                              or 3): "))
  if user_choice == 1:
      print(message, "eat")
  else:
      if user_choice == 2:
          print(message, "play")
      else:
          if user_choice == 3:
              print(message, "sleep")
          else:
              print("incorrect selection!")
```

```
Enter selection (1, 2, or 3): 2
Time to  play
```

Complete the function

Using nested `if` statements complete the `compare_nums1()` function which is passed two integers and returns a string. The function compares the first number to the second number and returns one of the following three strings (i.e., the string which is applicable):

"equal to" OR "less than" OR "greater than"

```
def compare_nums1(          ):





def main():
  num1 = random.randrange(1, 100)
  num2 = random.randrange(1, 100)
  comparison = compare_nums1(num1, num2)
  print(num1, "is", comparison, num2)
main()
```

Use a nested if to write the code

```
85 is greater than 21
```

```
64 is equal to 64
```

```
16 is less than 86
```

Python syntax of an if…elif statement

The **if…elif statement** allows at most one option (only one) to be executed out of many options. The else option (the last block) is optional.

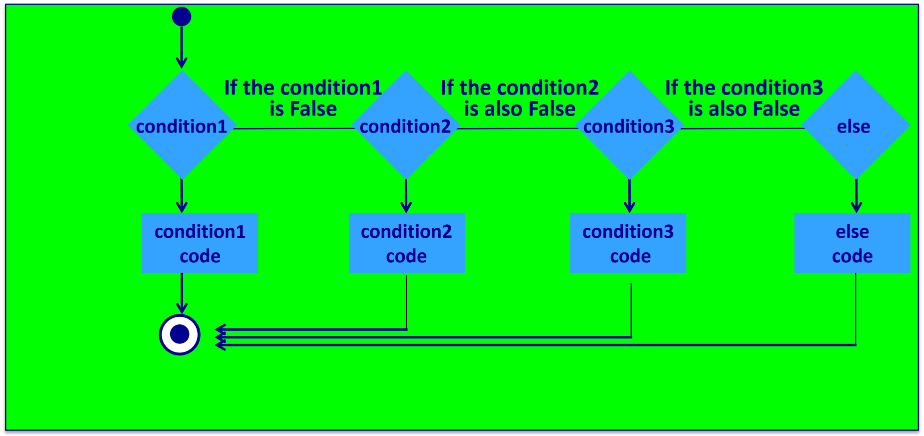
```
if boolean_expression1:
    statement1
    statement2
elif boolean_expression2:
    statement4
    statement5
elif boolean_expression3:
    statement6
    statement7
elif boolean_expression4:
    statement8
    statement9
else:
    statement10
    statement11
```

As soon as a match is found, the corresponding block of code is executed, then the if…elif statement is exited.

Note: at most one option is executed in an if…elif statement.

Python syntax for an if…elif statement

The following diagram shows an **`if…elif`** situation. As soon as a match is found, the corresponding block of code is executed, then the `if…elif` statement is exited.



condition1
If the condition1 is False
condition2
If the condition2 is also False
condition3
If the condition3 is also False
else
condition1 code
condition2 code
condition3 code
else code

Note: at most one option is executed in an if…elif statement.

An `if…elif` statement - example

A clearer way of writing the program from slide 10 is to use an **`if … elif`** statement:

```
def what_to_do_now():
  message = "Time to "
  prompt = "Enter selection (1, 2, or 3): "
  user_choice = int(input(prompt))

  if user_choice == 1:
      print(message, "eat")
  elif user_choice == 2:
      print(message, "play")
  elif user_choice == 3:
      print(message, "sleep")
  else:
      print("incorrect selection!")
```

```
Enter selection (1, 2, or 3): 2
Time to  play
```

Complete the function

Using and `if … elif` statement complete the `compare_nums2()` function which is passed two integers and returns a string. The function compares the first number to the second number and returns one of the following three strings (i.e., the string which is applicable):

"equal to" OR "less than" OR "greater than"

```
def compare_nums2(            ):




def main():
   num1 = random.randrange(1, 100)
   num2 = random.randrange(1, 100)
   comparison = compare_nums2(num1, num2)
   print(num1, "is", comparison, num2)
main()
```

Use an if…elif to write the code

```
16 is less than 86
```

```
64 is equal to 64
```

```
85 is greater than 21
```

Complete the function

A year is a leap year if it is divisible by 400, or divisible by 4 but not divisible by 100, e.g., 1900, 2011 and 2100 are not a leap years whereas 2000, 2008 and 2400 are leap years. Complete the `is_leap_year()` function.

```
def is_leap_year(year):




def main():
   print(is_leap_year(1900))
   print(is_leap_year(2011))
   print(is_leap_year(2100))
   print(is_leap_year(2000))
   print(is_leap_year(2008))
   print(is_leap_year(2018))
main()
```

```
False
False
False
True
True
False
```

The Python 'in' operator

The Python **'in' operator** can be used in boolean expressions to test if a string is part or all of another string.

```
def search_feedback(to_look_for, text_to_search):
   if to_look_for in text_to_search:
      print('It is there!')
   else:
      print('Not there!')

def main():
   search_feedback("messy","Embrace the glorious mess that you are")
   search_feedback("55 ","654 6557 999 555 ")

main()
```

```
Not there!
It is there!
```

If statements – exercise

Complete the `get_random_horoscope()` function which returns a random message. The function has 4 chances in 10 of returning "Amazing day ahead", 3 chances in 10 of returning "Romance is very likely", 1 chance in 10 of returning "Proceed with caution" and 2 chances in 10 of returning "Lucky lucky you".

```
import random
def get_random_horoscope():
   message1 = "Amazing day ahead"
   message2 = "Romance is very likely"
   message3 = "Proceed with caution"
   message4 = "Lucky lucky you"



def main():
   print("Today's message:", get_random_horoscope())
   print("Today's message:", get_random_horoscope())
main()
```

```
Today's message: Romance is very likely
Today's message: Amazing day ahead
```

get_random_horoscope() – solution 1

A solution to the function on slide 17:

```
def get_random_horoscope():
   message1 = "Amazing day ahead"
   message2 = "Romance is very likely"
   message3 = "Proceed with caution"
   message4 = "Lucky lucky you"
   message = ""
   number = random.randrange(0, 10)
   if number >= 0 and number < 4:
      message = message1
   if number >= 4 and number < 7:
      message = message2
   if number >= 7 and number < 8:
      message = message3
   if number >= 8 and number < 10:
      message = message4
   return message
```

get_random_horoscope() – solution 2

A second solution to the function on slide 17:

```
def get_random_horoscope():
   message1 = "Amazing day ahead"
   message2 = "Romance is very likely"
   message3 = "Proceed with caution"
   message4 = "Lucky lucky you"
   message = ""
   number = random.randrange(0, 10)
   if number < 4:
      message = message1
   elif number < 7:
      message = message2
   elif number < 8:
      message = message3
   else:
      message = message4
   return message
```

get_random_horoscope() function – solution 3

A third solution to the function on slide 17:

```
def get_random_horoscope():
  message1 = "Amazing day ahead"
  message2 = "Romance is very likely"
  message3 = "Proceed with caution"
  message4 = "Lucky lucky you"
  message = message4
  number = random.randrange(0, 10)

  if number < 4:
    message = message1
  elif number < 7:
    message = message2
  elif number < 8:
    message = message3

  return message
```

get_random_horoscope() – solution 4

A fourth solution to the function on slide 17:

```
def get_random_horoscope():
  message1 = "Amazing day ahead"
  message2 = "Romance is very likely"
  message3 = "Proceed with caution"
  message4 = "Lucky lucky you"

  number = random.randrange(0, 10)

  if number < 4:
    return message1
  elif number < 7:
    return message2
  elif number < 8:
    return message3
  else:
    return message4
```

get_random_horoscope() – solution 5

A fifth solution to the function on slide 17:

```
def get_random_horoscope():
  message1 = "Amazing day ahead"
  message2 = "Romance is very likely"
  message3 = "Proceed with caution"
  message4 = "Lucky lucky you"

  number = random.randrange(0, 10)

  if number < 4:
    return message1
  elif number < 7:
    return message2
  elif number < 8:
    return message3

  return message4
```

get_random_horoscope() – solution 6

A sixth solution to the function on slide 17:

```
def get_random_horoscope():
  message1 = "Amazing day ahead"
  message2 = "Romance is very likely"
  message3 = "Proceed with caution"
  message4 = "Lucky lucky you"

  number = random.randrange(0, 10)

  if number < 4:
    return message1
  if number < 7:
    return message2
  if number < 8:
    return message3

  return message4
```

get_random_horoscope() – OOOPS!

Why is the following code not a correct solution?

```
def get_random_horoscope():
  message1 = "Amazing day ahead"
  message2 = "Romance is very likely"
  message3 = "Proceed with caution"
  message4 = "Lucky lucky you"

  if random.randrange(0, 10) < 4:
    return message1
  elif random.randrange(0, 10) < 7:
    return message2
  elif random.randrange(0, 10) < 8:
    return message3

  return message4
```

Summary

In a Python program:

- the `if` block of an `if…else` statement is executed only if the boolean expression evaluates to `True`, otherwise the `else` block is executed.
- `if` statements can be nested inside other `if` statements.
- `if…elif…` statements are useful if there is a situation where at most one option is to be selected from many options. The `if…elif…` statement has an optional final `else` part.

Examples of Python features used in this lecture

```
if temperature > 25:
      print("Wear shorts.")
else:
      print("Not hot today!")
      print("Wear long pants.")

message = "Time to "
user_choice = int(input("Enter selection (1, 2, or 3): "))

if user_choice == 1:
      print(message, "eat")
elif user_choice == 2:
      print(message, "play")
elif user_choice == 3:
      print(message, "sleep")
else:
      print("incorrect selection!")
```