Chapter 2: Flow of Control

Last Update: January 31, 2023

Program 1: We show several expressions resulting in Boolean values. There is one grammatical error in this example.

02 11-boolean.py

```python
def next():
    input("Next> ")

x, y, z = 5, 2, 0
print("1.", x>9, y>0, z=='0')
next()
print("2.", x>9 and y>0)
next()
print("3.", x>9 or y>0)
next()
print("4.", not x>9)
next()
print("5.", x==y, x!=y, not x==y, )
next()
print("6.", x=5)
```

Program 2: We are testing several string functions that return a Boolean value in this program. The last example asks for input. Try to make it false.

```
02  12-boolean function.py
def next():
    input("Next> ")

str1 = "0123"
str2 = "Tier 1"
str3 = "UH"

print("1.", str1.isdigit())
next()
print("2.", str1.isalpha())
next()
print("3.", str2.isupper())
next()
print("4.", str2.isalnum())
next()
print("5.", str3.isalpha())
next()
print("6.", str3.isalnum())
next()
str4 = input("Try a string here: ")
print("7.", str4.isalnum())
```

Program 3: A number (integer or float) or a string can be cast into a Boolean type. It is easier to remember what will be cast into False because there are fewer. This program shows several examples.

```
02 13-cast boolean.py
```

```python
def next():
    input("Next> ")

print("1.", bool(1))
next()
print("2.", bool(-1))
next()
print("3.", bool(99))
next()
print("4.", bool(9.99))
next()
print("5.", bool("UH"))
next()
print("6.", bool("0"))
next()
print("7.", bool(" "))
next()
print("8.", bool(0))
next()
print("9.", bool(0.0))
next()
print("10.", bool(''))
```

Program 4: This program shows comparisons can be chained. The last example is a bad case.

```
02  14-chained comp.py
```

```python
def next():
    input("Next> ")

x = 15
if (x >= 10) and (x <= 20):
    print(x, "is inside 10 and 20." )
else:
    print(x, "is outside 10 and 20.")
next()

if (10 <= x <= 20):
    print(x, "is inside 10 and 20." )
else:
    print(x, "is outside 10 and 20.")
next()

if (100 >= x > 20):
    print(x, "is inside 100 and 20." )
else:
    print(x, "is outside 100 and 20.")
next()

if (10 >= x <= 20): # This works but BAD
    print(x, "is inside 10 and 20." )
else:
    print(x, "is outside 10 and 20. ???")
```

Program 5: There is a built-in function (no need to import anything) that asks if a given value is of a given type.

```
02  15-isinstance.py
def next():
    input("Next> ")

print("1.", isinstance(5,int))
next()

print("2.", isinstance(5.0,int))
next()

print("3.", isinstance('',str))
next()

print("4.", isinstance('a, b, c',list))
next()

x = '12345'
if isinstance(x, str):
    print("5.", 'True', x, type(x), id(x), len(x))
else:
    print("5.", 'False', x, type(x), id(x))
next()

x = 12345
if isinstance(x, str):
    print("6.", 'True', x, type(x), id(x), len(x))
else:
    print("6.", 'False', x, type(x), id(x))
```

Program 6: Testing the Boolean "in" operator.

```
02 16-boolean-in.py
```

```python
def next():
    input("Next> ")

print("1.", 5 in [1, 3, 5, 7, 9])
next()
print("2.", 6 in [1, 3, 5, 7, 9])
next()
print("3.", not 6 in [1, 3, 5, 7, 9]) # works
next()
print("4.", 6 not in [1, 3, 5, 7, 9]) # more natural
next()
print("5.", 'hell' in s)
next()
print("6.", 'he' in s)
next()
print("7.", 'ell' in s)
next()
print("8.", 'Hello' in s)
next()
print("9.", 'low' in s)
```

Program 7: Testing the Boolean "is" operator. The last example is tricky.

```
02 17-is.py
```

```python
# Identity operator
def next():
    input("Next> ")

s = "hello"
t = s
print ("1.", t is s)
print(f"t id={id(t)},  s id={id(s)}")
next()

t = 'or'
print ("2.", t is s)
print(f"t id={id(t)},  s id={id(s)}")
next()

s = [1, 2, 3]
t = [1, 2, 3]
print ("3.", t is s)
print(f"t id={id(t)},  s id={id(s)}")
next()

s = t = [5, 6, 7]
print ("4.", t is s)
print(f"t id={id(t)},  s id={id(s)}")
```

Program 8:

```
02 21-if.py
```

```
# Keywords: if-only, block structure
def next():
    input("Next> ")

x, y = 3, 5
if y**2 > 10:
    print("1. Yes!")
print("That's it.")
next()

if x**2 > 10:
    print("2. Yes!")
print("That's it.")
next()

if y**2 > 10:
    print("3. One!")
    print("3. Two!")
    print("3. Three!")
else:
    print("3. Four!")
    print("3. Five!")
    print("3. Six!")
print("That's it.")
```

Program 9: Testing if-else.

02 22-if else.py

```python
# Keywords: if-else
def next():
    input("Next> ")

x, y = 3, 5
if y**2>10:
    print("\tYes!")
    print("\tYes!")
    print("\tYes!")
else:
    print("\tNo!")
    print("\tNo!")
print("Back together.")
next()

if x**2>10:
    print("\tYes!")
else:
    print("\tNo!")
    print("\tNo!")
print("Back together.")
```

Program 10: This program shows how to use the else-if. Remove the #-sign in the last two lines and see what happens.

02 23-else-if.py

```python
def next():
    input("Next> ")

x = -9
# without using elif
if x == 1:
    print("\tOne")
else:
    if x < 0:
        print("\tTwo")
next()

# With elif
if   x == 1:
    print("\tOne")
elif x < 0:
    print("\tTwo")
next()

# Wrong way to use else if
if x == 1:
    print("\tOne")
#else if x < 0:
#    print("\tTwo")
```

Program 11: Nested if-else.

02 24-nested-it-else.py

```python
num = int(input("Enter a number: "))

if num<20:
    print("\t<20")
    if num<10:
        print("\t\t<10")
    else:
        print("\t\t>=10")
else:
    print("\t>=20")
    if num<30:
        print("\t\t<30")
    else:
        print("\t\t>=30")
```

Program 12: A better way for testing for multiple cases.

```
02 25-elif.py
def next():
    input("Next> ")

day = int(input("Enter a number 1-7: "))
if day==1:
    print("Monday")
else:
    if day==2:
        print("Tuesday")
    else:
        if day==3:
            print("Wednesday")
        else:
            if day == 4:
                print("Thursday")
            else:
                if day == 5:
                    print("Friday")
                else:
                    if day == 6:
                        print("Saturday")
                    else:
                        if day == 7:
                            print("Sunday")
                        else:
                            print("Invalid")
next()

# Symmetric, easier to understand
if   day==1:
    print("Monday")
elif day==2:
    print("Tuesday")
elif day==3:
    print("Wednesday")
elif day==4:
    print("Thursday")
elif day==5:
    print("Friday")
elif day==6:
    print("Saturday")
elif day==7:
    print("Sunday")
else:
    print("Invalid")
```

Program 13: The following is the first example of an unbalanced nested if-else structure. The two cases are different. The rule is to match the "else" with the "if" based on indentation.

```
02  26-nested-if.py
```

```
def next():
    input("Next> ")

a, b, c = 10, 20, 30
if (a<b):
    if (c<b):
        print("b is the max")
    else:
        print("b is the median")
# no outer-else
print("That's it")
next()

if (a<b):
    if (c<b):
        print("b is the max")
    # No inner-else
else:
    print("not sure")
print("That's it")
```

Program 14: Sometimes, it is easier to understand code with logical expression than if-else. In the example, we already have a Boolean value; why test it again?

```
02 27-test-vs-eval.py
```

```python
def next():
    input("Next> ")

test = bool(input("Enter a string:"))
print(test)
next()

if (test != True):
    result = True
else:
    result = False
print(result)
next()

# Here is an easier way
result = not test
print(result)
next()

# Even shorter, without using another variable
print(not test)
```

Program 15: In Case 1, the computer evaluated both expressions connected by an AND. Case 2 evaluates only the first one. If the second expression is evaluated, it will cause a "division by zero" error. Case 3 will cause the program to abort because it evaluates both expressions. There is no syntax error. The error is probably our first execution error.

```
02 31-short circuit.py
def next():
    input("Next> ")

x, y = 6, 2
if x>=2 and x/y>2: # True and True
    print("1. True", end="\n")
else:
    print("1. False", end="\n")
next()

x, y = 1, 0
if x>=2 and x/y>2: # False and ??? (Not executed)
    print("2. True", end="\n")
else:
    print("2. False", end="\n")
next()

x, y = 6, 0
if x>=2 and x/y>2:  # True and Error (Executed)
    print("3. True", end="\n")
else:
    print("3. False", end="\n")
```

Program 16: This example shows short circuits can happen to OR-ed expressions. In addition, it also demonstrates the side effect of evaluating an expression. Note that the function definition may be far from the if statement. Be mindful of the short circuits.

```
02 32-side effect.py
def test(n):
    print("*** Whatever ***")
    return bool(n%2)

x = int(input("Enter an int: ")) # Try 20, 5
if x >= 10 or test(x):
    print("Test: True")
else:
    print("Test: False")
```

Program 17: Computation vs. Comparison. The second case is much better than the first one. Case 3 is better if there is no need to remember the result.

02 33-simplify.py

```python
x, y = 10, 20
if x**2 > y:
    result = True
else:
    result = False
print('1.', result)

result = x**2 > y
print('2.', result)

print('3.', x**2 > y)
```

Program 18: Even though it is possible to squeeze multiple statements into one line, it is NOT recommended. Note that no "separator" is used in Case one, while Cases 2 and 3 use the separators.

02 41-if block.py

```python
def next():
    input('next> ')

status = int(input("Enter a number: "))
if status==1:
    print("1. Hello")
    print("world")
    print("!")
next()

# Yes, you can do it this way, BUT it is BAD
if status==1:
    print("2. Hello"); print("world"); print("!")
next()

# Even this
if status==1: print("3. Hello"); print("world"); print("!")
```

Program 19: Case one put the final print outside the if-elif, while Case 2 put it inside the if-elif. We are using "tab" to show the difference.

```
02 43-nested elif.py
# Try 1, 2, 3, 4
def next():
    input('Next> ')

status = int(input("Enter a number: "))
if   status==1:
    print("\t1. One")
elif status==2:
    print("\t1. Two")
elif status==3:  # no else
    print("\t1. Three")
print("1. That's all.")
next()

if   status==1:
    print("\t2. One")
elif status==2:
    print("\t2. Two")
elif status==3:  # else?
    print("\t2. Three")
else:
    print("\t2. That's all.")
```

Program 20: This program shows a 3-way if-else structure. The three cases here happen to be mutually exclusive. Not all such structures are mutually exclusive. Continue to the following example.

```
02 51-elif.py
x = int(input("Please enter an integer: "))
if x<0:
    print(f'{x} is negative')
elif x==0:
    print(f'{x} is zero')
else:
    print(f'{x} is positive')
print('Job done.')
```

Program 21: The two cases are equivalent because they are mutually exclusive. What happens if it is not? Give an example.

```
02 52-elif.py
def next():
    input('next> ')

num1 = int(input("Enter the first number: "))
num2 = int(input("Enter the second number: "))
next()
# Case 1
if   num1 < num2:
    print(num1, "is smaller.")
elif num1 > num2:
    print(num2, "is smaller.")
else:
    print("The two numbers are equal.")
next()
# Case 2
if num1 < num2:
    print(num1, "is smaller.")
if num1 > num2:
    print(num2, "is smaller.")
if num1 == num2:
    print("The two numbers are equal.")
```

Program 22: This program is for comparison with the following program.

```
02  60-if else.py
def next():
    input('next> ')

age = eval(input("Enter your age: "))
if age < 18:
    if age < 12:
        print("Kid")
    else:
        print("Teeager")
else:
    print("Adult")
next()

# A better arrangement; simplicity
if   age < 12:
    print("Kid")
elif age < 18:
    print("Teeager")
elif age < 65:
    print("Adult")
else: # Over 65
    print("Senior")
```

Program 23: Using conditional expression.

```
02  61-cond exp.py
def next():
    input('next> ')

age = eval(input("Enter your age: "))
print("kid"       if age<12 else
      "teenager" if age<18 else
      "adult"    if age<65 else
      "senior")
next()

status = ("kid" if age<12 else
        "adult" if age<18 else
        "adult" if age<65 else
        "senior")
print(status)
```

Program 24: Using the conditional expression for numerical computation.

```
02 62-cond exp.py
num = eval(input("enter a number: "))
abs = num if num>=0 else -num
print ("The absolute value of", num, "is", abs)
```

Program 25: Conditional expression is shorter and better than if-else.

```
02 63-cond exp.py
def next():
    input('next> ')

x = int(input("Enter an integer number: "))

if x == 1:
    s = 'a'
elif x == 2:
    s = 'b'
elif x == 3:
    s = 'c'
else:
    s = 'd'
print(s)
next()

s = ('a' if x == 1 else
     'b' if x == 2 else
     'c' if x == 3 else
     'd'
    )
print(s)
next()

# Shorter, symmetric, and simpler
print('a' if x == 1 else
      'b' if x == 2 else
      'c' if x == 3 else
      'd'
    )
```

Program 26: One more example of simplification.

```
02 64-cond exp.py
def next():
    input("Next> ")

food = input("What is your favorite food? ")

if food == "lamb":
    reply = "Yuck"
else:
    reply = "Yum"
print(food, reply)
next()

reply = "Yuck" if food=="lamb" else "Yum"
print(food, reply)
next()

print(food, "Yuck" if food=="lamb" else "Yum")
```