Control Statements

Exercises

Week 3

Prior to attempting these exercises ensure you have read the lecture notes and/or viewed the video, and followed the practical. You may wish to use the Python interpreter in interactive mode to help work out the solutions to some of the questions.

Download and store this document within your own filespace, so the contents can be edited. You will be able to refer to it during the test in Week 6.

Enter your answers directly into the highlighted boxes.

For more information about the module delivery, assessment and feedback please refer to the module within the MyBeckett portal.

What is the *data-type* of the result when evaluating comparison (relational) expressions such as < and >?

Answer:

The result of a comparison (relational) expression, such as < (less than) or > (greater than), is a Boolean value.

For each of the following expressions write the result of their evaluation.

```
100 < 101
```

Answer:

True

```
100 > 99
```

Answer:

True

```
100 >= 100
```

Answer:

True

```
100 != 100
```

Answer:

False

For each of the following expressions write the result of their evaluation.

```
"abc" < "xyz"
```

Answer:

True

```
"abc" < "XYZ"
```

Answer:

False

```
"100" == 100
```

Answer:

False

For each of the following expressions write the result of their evaluation.

```
10 > 20 and 10 >= 10
```

Answer:

10 > 20: This is false because 10 is not greater than 20.
10 >= 10: This is true because 10 is greater than or equal to 10.

false and true: This evaluates to false because both conditions must be true for the entire expression to be true.
Therefore, the result of the expression 10 > 20 and 10 >= 10 is false.

```
10 > 30 > 20
```

Answer:

10 > 30: This is false because 10 is not greater than 30.
30 > 20: This is true because 30 is greater than 20.
Now, the full expression is a combination of these two conditions:

false and true: This evaluates to false because both conditions must be true for the entire expression to be true.
Therefore, the correct result of the expression 10 > 30 > 20 is false

```
40 < 20 or 20 < 30
```

Answer:

40 < 20: This is false because 40 is not less than 20.
20 < 30: This is true because 20 is less than 30.
Now, the full expression is a combination of these two conditions using the or operator:

false or true: This evaluates to true because at least one of the conditions is true.
Therefore, the result of the expression 40 < 20 or 20 < 30 is true.

```
not True
```

Answer:

not True: This evaluates to false because it negates the truth value of True.
Therefore, the result of the expression not True is false.

What would be the output shown following the execution of the following Python statements?

```
colours = [ "Blue", "Black", "Orange" ]

print("The colour black is in the list : ", "Black" in colours)
```

Answer:

The colour black is in the list : True

```
print("The colour orange is in the list : ", "orange" in colours)
```

Answer:

The colour orange is in the list : False

Which of the following concepts does the Python '`if`' statement support?

Sequence, **Selection** or **Iteration**?

Answer:

The Python 'if' statement supports the concept of "Selection." The 'if' statement is a conditional statement that allows you to execute a block of code based on whether a specified condition is true or false. It enables you to make decisions in your code, choosing different paths of execution based on the evaluation of a given condition. "Selection" refers to the process of selecting a particular block of code to be executed based on the outcome of a condition, and the 'if' statement is a key element in implementing this concept in Python.

What would be the output shown following the execution of the following Python statements?

```
num1 = 100
num2 = 10

if num1 % num2 == 0:
      print("num1 is divisible by num2")
else:
      print("num1 is not divisible by num2")
```

Answer:

num1 is divisible by num2

What would be the output shown following the execution of the following Python statements?

```
num1 = 99
num2 = 70

if num1 < num2:
      print("num1 is less than num2")
elif num1 > num2:
      print("num1 is greater than num2")
else:
      print("num1 is equal to num2")
```

Answer:

num1 is greater than num2

What is the name given to the following type of Python operator shown below?

```
lowest = x if x < y else y
```

Answer:

Ternary Operator

And, what value would be assigned to the variable '`lowest`' when '`x`' was equal to `10` and '`y`' was equal to `5`?

Answer:

lowest = x if x < y else y

Within the answer box below write a small Python program, that asks the user to enter a value between 1 and 10.

Once the value has been input display a message saying whether the value was in the requested range.

Remember: values returned from the **input()** function are *strings*, and need converting before being used within expressions, i.e. you will need code such as this -

```
num = input("please enter a number between 1 and 10 : ")
num = int(num)
```

Answer:

```
num = input("Please enter a number between 1 and 10: ")
num = int(num)
if 1 <= num <= 10:
```

```
print("The entered value is in the requested range.")
else:
print("The entered value is outside the requested range.")
```

Within the answer box below write a small Python program that asks the user to enter two values. Store these in variables called `x` and `y` respectively.

If the 'x' value is larger than 'y' then print

```
The value 'x' is larger than the value 'y'
```

otherwise print

```
The value 'y' is larger than the value 'x'
```

Answer:

```
x = float(input("Enter the first value (x): "))
y = float(input("Enter the second value (y): "))

if x > y:
print(f"The value {x} is larger than the value {y}.")
else:
print(f"The value {y} is larger than the value {x}.")
```

Examine the output generated by the above program. Is the displayed text entirely accurate in all cases? If not Why?

Answer:

The displayed text is accurate for numeric input, but it doesn't handle non-numeric input or explicitly address the case of equal values. The modified version includes additional checks for these scenarios, providing a more robust program.

Within the answer box below write a small Python program that asks the user to enter two values.

Store these values in two variables then output a message displaying the result of dividing the first value by the second value.

Include code that prevents a run-time error being reported when the user inputs a value of '`0`' for the second input. *Hint:* use an '`if`' statement

If a '`0`' value is input, print a message saying "`division by 0 is not possible`".

Answer:

```
x = float(input("Enter the first value: "))
y = float(input("Enter the second value: "))
except ValueError:
print("Please enter valid numeric values.")
exit()

if y == 0:
print("Division by 0 is not possible.")
else:
result = x / y
print(f"The result of {x} divided by {y} is: {result}")
```

Which of the following concepts does the Python `while` statement support?

Sequence, **Selection** or **Iteration**?

Answer:

Iteration

What would be the output shown following the execution of the following Python statements?

```
num = 5

while num > 0:
      print(num)
      num -= 1
```

Answer:

5
4
3
2
1

Write a small Python program that prints your name to the screen 100 times, then enter the program into the answer box below. Hint: use a '`while`' loop.

Answer:

```
count = 0

while count < 100:
print("Your Name")
count += 1
```

What would be the output shown following the execution of the following Python statements?

```
vals = ["A", "B", "C", "D"]

for letter in vals:
      print(letter)
```

Answer:

A
B

C
D

What would be the output shown following the execution of each of the following Python statements?

```
for num in range(5):
      print(num)
```

Answer:

```
0
1
2
3
4
```

```
for num in range(10,16):
      print(num)
```

Answer:

```
10
11
12
13
14
15
```

```
for num in range(0,10,-1):
      print(num)
```

Answer:

```
10
9
8
7
6
5
4
3
2
1
```

Enter and execute the python code shown below, then show the exact output into the answer box.

```
for x in range(1,10):
      for y in range (1,x):
            print("*")
      print()
```

Answer:

```
*
*
*
*
*
*
*
*
*
*
*
*
*
*
*
*
*
*
*
```

What is the term used to refer to code blocks that appear inside other code blocks as in the above program?

Answer:

The term used to refer to code blocks that appear inside other code blocks is "nested" code blocks.

Exercises are complete

Save this logbook with your answers. Then ask your tutor to check your responses to each question.