Objective 4:
Understand how to use selection statements

In this objective you learn how to make different parts of your program run depending on the value of variables.

Tasks

1. Try entering the following commands and see what happens:

```
#Using selection statements to check variables
#Ask user for the number
number = int(input("Enter a number 1-3:"))
#Check the value of the number
if number == 1: print("Number one")
if number == 2: print("Number two")
if number == 3: print("Number three")
```

   == allows you to check if a variable equals a value.

2. Try entering the following commands and run the program three times with the test data: 6, 51 and 70.

   Pay careful attention to the indentation with tabs, these are not spaces.

```
      #Works out whether a candidate passed or failed
      score = int(input("Enter score:"))
      if score > 40:
TAB  →        print("You passed")
      else:
TAB  →        print("You failed")
```

   Note how the program only executes instructions if the condition is true i.e. the score is more than 40, and how it is possible to use an else statement to run an alternative set of commands if the condition is false.

3. Change the program so that you have to score over 50 to pass.

4. Try entering the following commands and run the program three times with the test data:
   6 & 4, 51 & 51.

```
#Returns whether two numbers are the same
num1 = int(input("Enter a number: "))
num2 = int(input("Enter a number: "))
if num1 != num2:
      print("The numbers are not the same")

else:
      print("The numbers are the same")
```

Note != means does not equal.

5. Try entering the following commands and run the program three times with the test data:
   1, 3, 5.

```
#Using logic operators
choice = input("Enter a number 1-3: ")
if choice > "0" and choice < "3":
    print("Valid input")
else:
    print("Invalid input")
```

Note how ‘and’ can be used to check if both conditions are true. You can also use ‘or’ if only one condition needs to be true and ‘not’ as an alternative to does not equal.

Note how the number was input as a string, not an integer, yet greater than and less than operators still work to check if the input was between 1 and 2. That’s because the ASCII value of the character is being checked, not the actual number. This approach is helpful as it prevents the program crashing if a character is input instead of a number.

6. Try entering the following commands and see what happens:

```
#Using case selection
#Ask user for the number
print("1. Add numbers")
print("2. Subtract numbers")
print("3. Quit")

choice=input("Enter your choice: ")
#Multiple branches depending on selection
if choice == "1":
    print("Add numbers chosen")
elif choice == "2":
    print("Subtract numbers chosen")
elif choice == "3":
    print("Quit chosen")
```

Note how it is possible to use `elif` to have multiple branches rather than just true or false.

Objective 4: Key learning points

How to use selection statements

- Checking the value of a variable and executing instructions depending on the outcome of the check is known as a **selection**.
- The instructions that are executed as a result of a selection is known as a **program branch**.
- The logical checking of a variable against another value is known as a **condition**.
- Elif commands can be used instead of multiple if statements.
- Code for each program branch must be indented.
- It is good practice to comment a selection to explain its purpose.
- One ‘If’ can be placed inside another ‘If’, this is known as **nesting**.

Objective 4: Key words

If… else: …

```
Example code:  If (x>0 and x<5) or x=10:
                     …
               else:
                     …
```

x is the variable being checked. Multiple variables can be checked in one condition. Use brackets to express order of priority. In the example code, x must be between 0 and 5 or equal to 10 for the condition to be true. Else is optional and is used to branch if the condition is false. All program branches must be indented.

Logical operators that can be used in conditions include:

==	equals	<	less than	and	both conditions must be true
!=	does not equal	<=	less than or equal to	or	one condition must be true
		>	greater than	not	condition is not true
		>=	greater than or equal to		

elif

```
Example code:  if x == 3:
                     …
               elif x < 3:
                     …
               elif x >= 10 and x <= 20:
                     …
               else:
                     …
```

x is the variable being checked. elif is used as an alternative to multiple if statements.