Programming Fundamentals

2022-batch BS (CySec)

Course teacher:	Dr. Muhammad Awais, Assistant professor, Department of Cyber Security
Course website:	https://tinyurl.com/22cys-pf
Email:	awaisrajput@quest.edu.pk engr.awais.rajput@gmail.com

Chapter -2

- Introduction to algorithms and flowcharts
- Basic data types and variables
- Input/output constructs
- Arithmetic operators
- Comparison and logical operators
- Conditional statements and execution flow for conditional statements

Algorithms

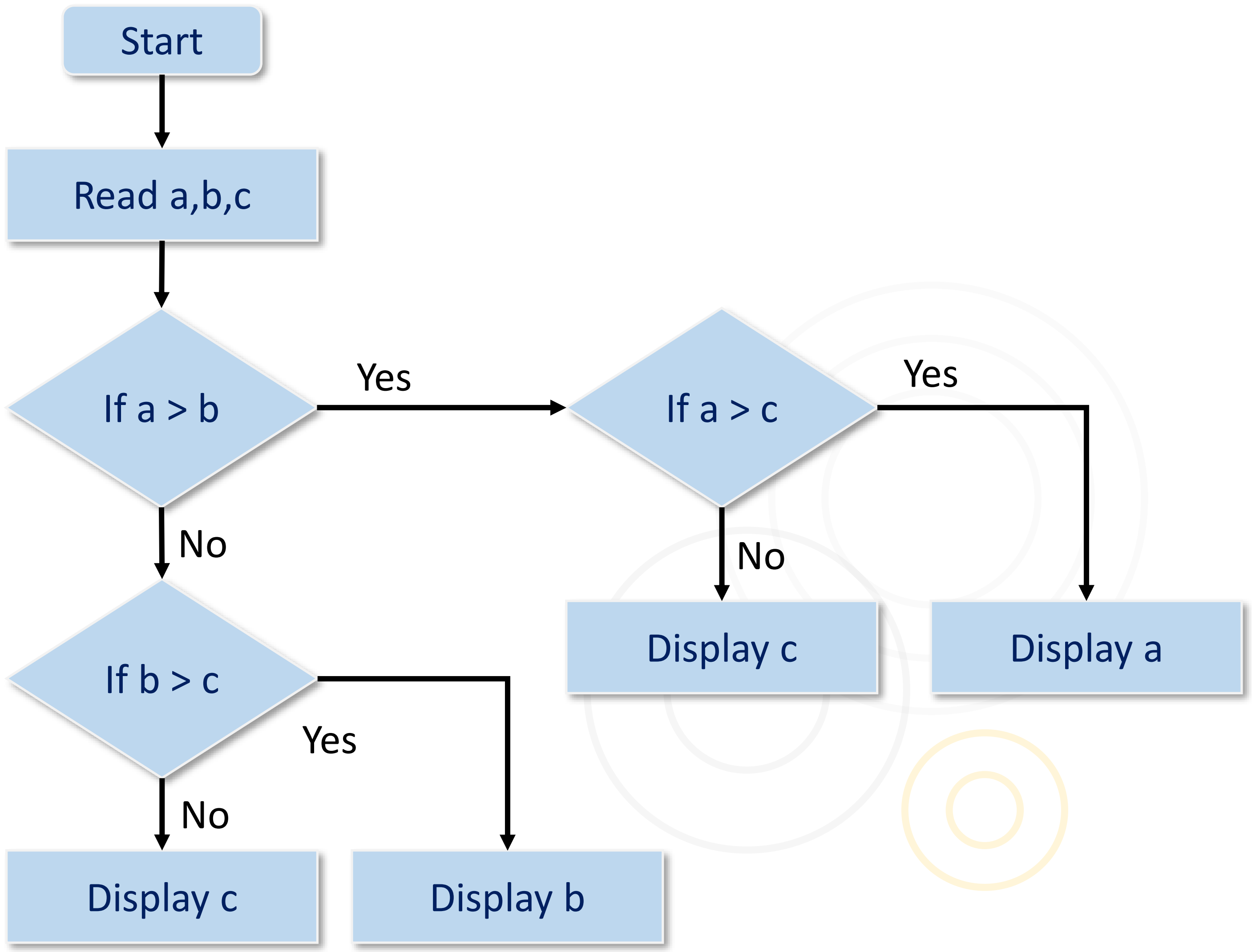
1. Preheat the oven
2. Gather the ingredients
3. Measure out the ingredients
4. Mix the ingredients to make the batter
5. Grease a pan
6. Pour the batter into the pan
7. Put the pan in the oven
8. Set a timer
9. When the timer goes off, take the pan out of the oven

```
1. Start
2. Declare variables num1, num2 and sum.
3. Read values num1 and num2.
4. Add num1 and num2 and assign the result to sum.
5.        sum←num1+num2
6. Display sum
7. Stop
```

An algorithm is a set of instruction written to solve any problem

Flowcharts – a graphical interpretation

```
1: Start
2: Declare variables a,b and c.
3: Read variables a,b and c.
4: If a > b
      If a > c
            Display a is the largest number.
      Else
            Display c is the largest number.
   Else
         If b > c
            Display b is the largest number.
         Else
            Display c is the greatest number.
5: Stop
```

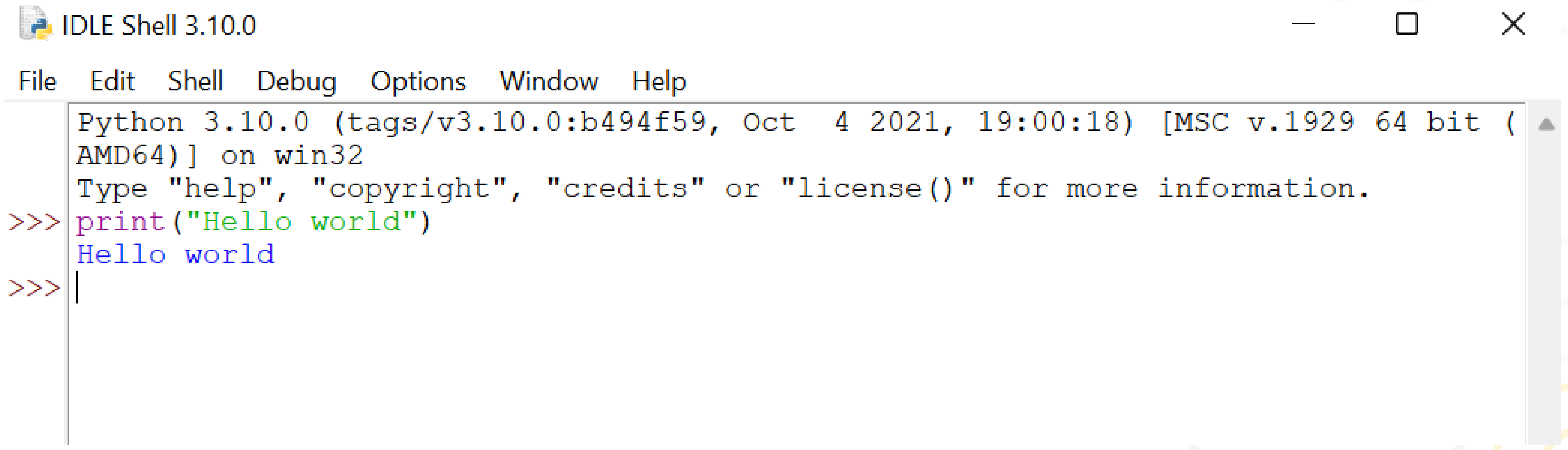


Introduction to Python

- Python is an interpreted, object-oriented, high-level programming language
- We will use Python to learn about programming basics because
  - It is one of the most popular languages out there
  - Easier to learn because of higher abstraction
  - Large community
  - Widely used in AI domain
- What you can use Python for
  - build websites and software
  - automate tasks
  - conduct data analysis
- How to get started?
  - Install Python and work with IDLE (Windows)

Very first program in Python

- Open IDLE
- Type
  - `print(“Hello world”)`
- And press Enter



```
Python 3.10.0 (tags/v3.10.0:b494f59, Oct  4 2021, 19:00:18) [MSC v.1929 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> print("Hello world")
Hello world
>>> 
```

Writing scripts in files

- Option -1
  - Open IDLE and create a new file
  - Write Python code, save it, and then choose „Run“
- Option -2
  - Open a text editor of your choice
  - Write Python code and save it as „.py“ file
  - Open command prompt, navigate to the file location and then issue „python <file_name>“ command

Data types

- Let's start modifying the first program

```
message = "Hello Python world!"
print(message)
```

- Write the above in a new script file and save it as “hello_world.py”
- Run the program and monitor the output
- We have just declared a variable (named *message*) and assigned it a value
- Further modifying the program

```
message = "Hello Python world!"
print(message)

message = "Hello 22 AI batch!"
print(message)
```

Data types (contd.)

- Variables allow you to keep values of different types e.g., integer, real, string
  - Examples: 10, 65.7, “Artificial Intelligence”
- There are some rules to keep in mind when using variables in Python
  1. Variable names can contain only letters, numbers, and underscores. They can start with a letter or an underscore, but not with a number.
  2. Spaces are not allowed in variable names, but underscores can be used to separate words in variable names.
  3. Avoid using Python keywords and function names as variable names.
  4. Variable names should be short but descriptive.
  5. Be careful when using the lowercase letter l and the uppercase letter O because they could be confused with the numbers 1 and 0.

What could happen if you break rule 1, 2, or 3?

Identify valid variable names: Age, 1age, _age, name of student, s_no#, _salary

Python keywords

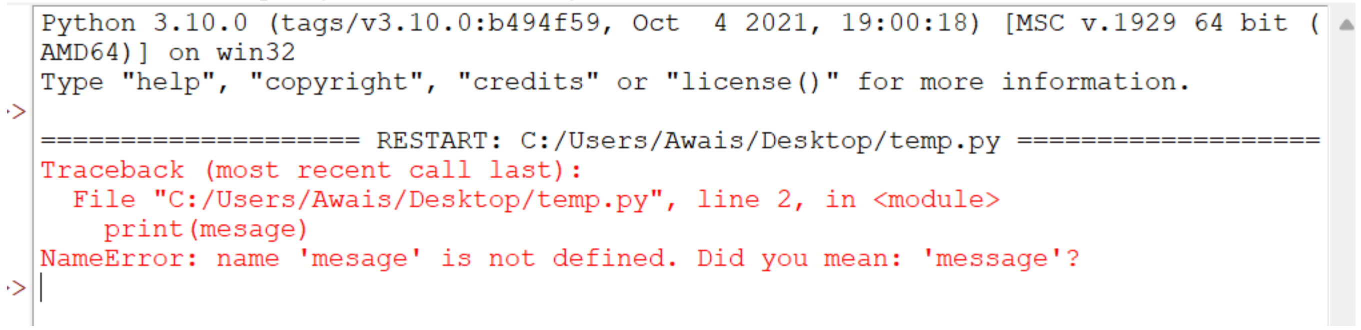
```
False      await      else       import     pass
None       break      except     in         raise
True       class      finally    is         return
and        continue   for        lambda     try
as         def        from       nonlocal   while
assert     del        global     not        with
async      elif       if         or         yield
```

https://docs.python.org/3/reference/lexical_analysis.html#keywords

Data types (contd.)

- What happens if you run the following program?

```
message = "Hello Python world!"
print(mesage)
```



```
Python 3.10.0 (tags/v3.10.0:b494f59, Oct  4 2021, 19:00:18) [MSC v.1929 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>
=================== RESTART: C:/Users/Awais/Desktop/temp.py ==================
Traceback (most recent call last):
  File "C:/Users/Awais/Desktop/temp.py", line 2, in <module>
    print(mesage)
NameError: name 'mesage' is not defined. Did you mean: 'message'?
>
```

Data types (contd.)

- String
  - Hold textual data i.e., series of characters
- Examples

```
"This is a string."
'This is also a string.'
```

- Common operations on string
  - Changing case (name.title(), name.upper(), name.lower())
  - Concatenation (the '+' operator, the f"{} operator, the format() function)
  - Trim (rstrip(), lstrip(), and strip(), Remember that the functions do not modify variable's value!!!)
  - Formatting with tab and new line ("\n", "\t")

Avoid mixing single and double quotations while using strings

Data types (contd.)

- Numbers
  - Hold numerical values
  - Examples: integers and real numbers (floats)
- Exercise: Perform some common math operations on numbers in IDLE terminal
- What happens if you mix integers and floats (Look for division case)?
- Multiple assignments of variables

```
>>> x, y, z = 0, 0, 0
```

- Constants

```
MAX_CONNECTIONS = 5000
```

Can you combine strings with numerical values to form output?

Data types - summary

- Basic data types in Python: integer, float, strings
- Some other are to be introduced later (bool, list, dict, etc.,)
- Variables hold values during the execution of the program
- Strings hold character sequences
- Integer and float contain integer and real data
- Rules to remember when naming variables

Comments

- Extremely useful when it comes to readability of the code
- Allows you or any other programmer understand the code (yes, you too!)
- Python uses '#' sign to denote comments, the interpreter ignores the comments in a code snippet
- Tips
    - Do not replicate the code (or similar) in the comment
    - Write clear, concise comments
    - Explain why you did it this way

Tasks

- Determine the type of a variable (hint: type())
- Setting specific data types (str(), float(), int())
- Generate random numbers (random.random(), random.randrange())
- Using strings as arrays (str[1])
- Using len() function
- Slicing strings (str[2:5]), from the start, from the end, negative indexing
- Replacing in string (name.replace())
- String concatenation
- Using input() function

Arithmetic operations

- Python suppoorts following arithmetic operations
  - Addition (+)
  - Subtraction (-)
  - Multiplication (*)
  - Division (/)
  - Modulus (%)
  - Exponentiation (**)
  - Floor division (//)
- Additionally, the assignment operator
  - =
  - +=
  - -=
  - *=
  - and so on…

Operator	Description	Syntax
+	Addition: adds two operands	x + y
–	Subtraction: subtracts two operands	x – y
*	Multiplication: multiplies two operands	x * y
/	Division (float): divides the first operand by the second	x / y
//	Division (floor): divides the first operand by the second	x // y
%	Modulus: returns the remainder when first operand is divided by the second	x % y
**	Power : Returns first raised to power second	x ** y

Arithmetic operations (contd.)

```
number = 1 + 2 * 3 / 4.0
print(number)
```

```
remainder = 11 % 3
print(remainder)
```

```
a = 7 ** 2
b = 2 ** 3
print(a)
print(b)
```

```
x = 6 / 2 + 1 - 3 + 8 * 4
print(x)
```

```
x = 6 / (2 + 1) - (3 + 8) * 4

print(x)
```

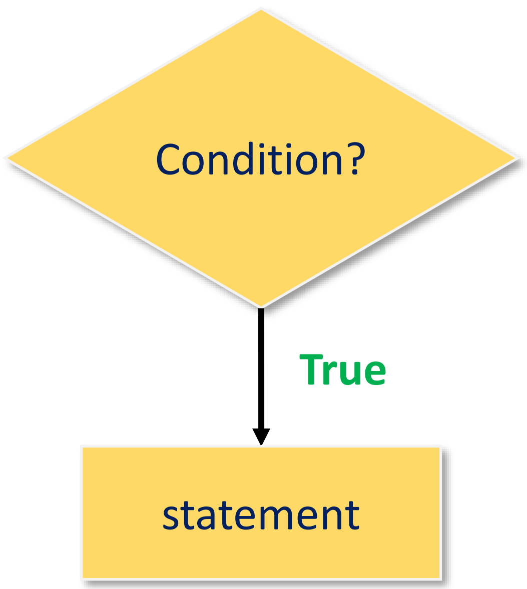
Tasks – arithmetic operations

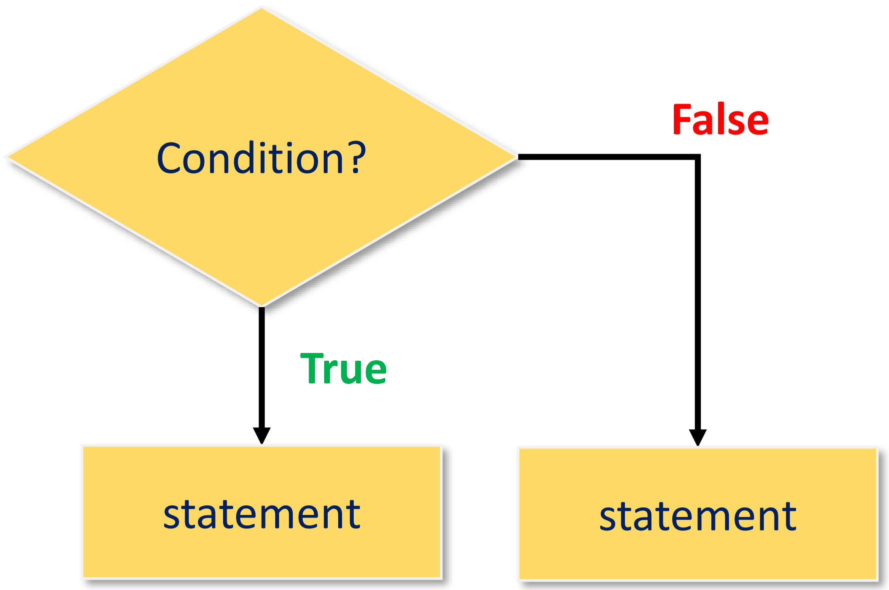
- Write a program that inputs your age in years and then calculate the total number of months, weeks and days.
- Write a program that inputs marks for 5 subjects and then calculate the total marks obtained and the average
- Write a program that takes temperature in Fahrenheit as input and convert it into Celsius. (hint: C= (F − 32) * 5/9)
- Write a program that takes input the coefficients (a,b,c) of a quadratic equation and calculate its roots x1 and x2 with the following formulae:

$$x_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a} \quad \text{and} \quad x_2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a}$$

Conditional statement

- Real life example
  - If it rains, I will stay home else I can go shopping
  - If I pass this course, I will get a certificate
- Python supports conditional statements with ‘if’, ‘else’ blocks
- What would be the output of the code snippets?





```
a = 33
b = 200
if b > a:
  print("b is greater than a")
```

```
a = 33
b = 33
if b > a:
  print("b is greater than a")
elif a == b:
  print("a and b are equal")
```

```
a = 200
b = 33
if b > a:
  print("b is greater than a")
elif a == b:
  print("a and b are equal")
else:
  print("a is greater than b")
```

Comparison and logical operators

Comparison operators		
Operator	Name	Example
==	Equal	x == y
!=	Not equal	x != y
>	Greater than	x > y
<	Less than	x < y
>=	Greater than or equal to	x >= y
<=	Less than or equal to	x <= y

Logical operators		
Operator	Name	Example
and	Returns True if both statements are true	x < 5 and x < 10
or	Returns True if one of the statements is true	x < 5 or x < 4
not	Reverse the result, returns False if the result is true	not(x < 5 and x < 10)

Python if-else statement

- Simple *if*

```
if condition:
   # Statements to execute if
   # condition is true
```

- *If-else*

```
if (condition):
    # Executes this block if
    # condition is true
else:
    # Executes this block if
    # condition is false
```

- Nested *if-else*

```
if (condition1):
   # Executes when condition1 is true
   if (condition2):
      # Executes when condition2 is true
   # if Block is end here
# if Block is end here
```

Python if-else statement (contd.)

- *If-elif-else* ladder

```
if (condition):
    statement
elif (condition):
    statement
.
.
else:
    statement
```

Python if-else statement (contd.)

- Examples

```
i = 10

if (i > 15):
        print("10 is less than 15")
print("I am Not in if")
```

```
i = 20
if (i < 15):
    print("i is smaller than 15")
    print("i'm in if Block")
else:
    print("i is greater than 15")
    print("i'm in else Block")
print("i'm not in if and not in else Block")
```

Tasks – if-else

- A user inputs a number. The program determines whether the number is positive, negative or 0
- A user inputs the current temperature. The program shows the appropriate message
- A user inputs a year, and the program determines whether it is a leap year
- A company decided to give bonus of 5% to employee if his/her year of service is more than 5 and his/her age is >35 years. Ask user for their salary, age and year of service and print the net bonus amount.
- Take values of length and breadth of a rectangle from user and check if it is square or not.
- Take three int values from user and print greatest among them.
- A shop will give discount of 10% if the cost of purchased quantity is more than 1000. Ask user for quantity. Suppose, one unit will cost 100. Compute and print total cost for user.

How to know if it is a Leap Year:
Can be exactly divided by 4 BUT cannot be exactly divided by 100
except if it can be exactly divided by 400

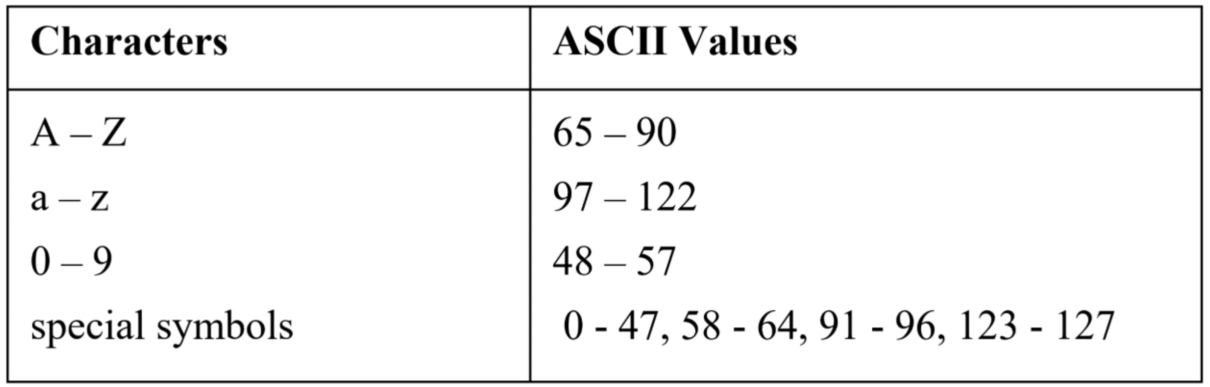
Tasks (contd.)

- Write a program to print absolute value of a number entered by user.
- A student will not be allowed to sit in exam if his/her attendance is less than 75%. Take following input from user: number of classes held, number of classes attended, and print percentage of class attended and whether student is allowed to sit in exam or not.
- Write a program to calculate the electricity bill (accept number of unit from user) according to the following criteria :

Unit	Price
First 100 units	no charge
Next 100 units	Rs 5 per unit + Rs. 100 surcharge
After 200 units	Rs 10 per unit + Rs. 200 surcharge

Tasks (contd.)

- A user inputs his obtained marks. The program has to calculate his grade according to the following rules:
    1. <40: Fail
    2. 41-50: ‘D’ grade
    3. 51-60: ‘C’ grade
    4. 61-70: ‘B’ grade
    5. 71-80: ‘A’ grade
    6. >81: ‘A-1’ grade



Characters	**ASCII Values**
A – Z	65 – 90
a – z	97 – 122
0 – 9	48 – 57
special symbols	0 - 47, 58 - 64, 91 - 96, 123 - 127

- Write a program that asks for a value and then determines whether the entered value was a number or a character.

Summary

- Algorithms and flowcharts enable easier comprehension of the code
- Basic arithmetic operations in Python and their order
- Logical operators, comparison operators in Python
- Conditional statements and examples

References and acknowledgment

- Some of the material is taken from the slides of Dr Umair Ali Khan
- Books
  - Python crash course: a hands-on, project-based introduction to programming by Eric Matthes
- Online resources
  - https://docs.python.org/3/tutorial
  - https://www.geeksforgeeks.org
  - https://www.w3schools.com/python
  - https://www.codesdope.com/