Python Course Summarization Part 1

made by Rawan Hatem

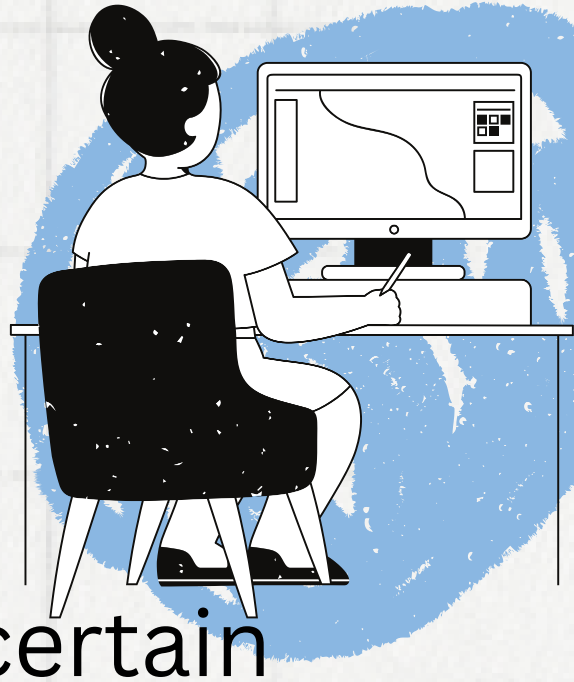
variables and data types

- Variables in Python are reusable containers for storing values.
- Variables should have descriptive and unique names.
- To print a variable, it can be placed within a print statement without quotes.
- Concatenation is used to display a variable along with some text.
- To concatenate, a variable should be separated from the text with the "+" symbol.
- If a number is to be displayed along with text, typecasting is required to change the variable into a string using the "str()" function.
- Spacing is important while displaying variables along with some text.
- There are four basic data types in Python: integers, floats, strings, and booleans.

Math in Python

- Built-in math-related functions:
  - Round function (result = round(x))
  - Absolute value function (result = abs(y))
  - Power function (result = pow(y, 3))
  - Maximum ,Minimum value function (result = max(x, y, z)),(result=min(x,y,z)
- constants and functions from the math class , requiring the import of the math module:
  - Value of pi (**math.pi**)
  - Exponential constant **e** (**math.e**)
  - Square root function (**result = math.sqrt(x)**)
  - Ceiling function (**result = math.ceil(x)**)

If, elif, else statements



- If statements in programming are used for decision making based on a certain condition.
- If the condition is true, any code found within the if statement will be executed. If the condition is false, no code will be executed.
- An else statement can be added to the if statement to specify code that should be executed if the condition is false.
- An elif (else if) statement can also be added to check for additional conditions before reaching the else statement.
- When checking for equality in if statements, use double equals (==) instead of a single equals sign (=), which is used for assignment.
- If statements can also use boolean variables directly as conditions, without needing to write out a full condition.

Logical Operators

- Logical operators are used in conditional statements, such as "if" statement.
- There are three types of logical operators: "and", "or", and "not".
- "And" exhibits behavior where two or more conditions must be true.
- "Or" exhibits behavior where at least one condition must be true.
- "Not" changes the result of a condition; if a condition is true, it becomes false, and if it was originally false, it becomes true.

while loops

- A while loop executes code while a certain condition remains true.
- Without a way to exit the while loop, it can lead to an infinite loop
- A while loop can also use logical operators such as "not" and "or" to control the execution of the code.

For Loop

- For loops can iterate over a range, a string, or any iterable sequence.
- For loops are ideal for situations where you need to do something a fixed number of times.
- Basic syntax for a for loop: for <counter> in range(<start>, <stop>):, where <stop> is exclusive.
- To count backwards, you can use the reversed() function: for <counter> in reversed(range(<start>, <stop>))
- The range() function has an additional parameter, step, to count by twos, threes, etc.
- Besides **range()**, you can also iterate over a string
- Useful keywords for for loops:
  - continue: skips the current iteration when a certain condition is met.
  - break: breaks out of the loop entirely when a certain condition is met.
- For loops are often interchangeable with while loops, but while loops are better when executing something possibly an infinite number of times, like accepting user input.

Nested Loop

- Nested loops refer to a loop within the code of another loop.
- The loop on the outside is called the outer loop, and the internal loop is called the inner loop.
- The outer loop controls the rows, and the inner loop controls the columns.
- Various combinations of while and for loops can create nested loops.
- A print statement ends with a new line character by default, but it can be changed to a comma or another character by add end parameter ->print(x ,end=" ")
- When using nested loops, make sure the counters have different names.
-

Methods to Check Palindrome in Python

01.

Using the reverse and compare method

- #Enter input string
  - string = input ("Enter string: ")
- look for the reversed string and then compare it to the original string
  - if (string == string[::-1]) :

    print("The string is a palindrome.")
  - else:

    print("The string is not a palindrome.")

02.

The predefined function "".join(reversed(string))" is used in this method to reverse the string.

# function to check string is palindrome or not

# Using predefined function to reverse to string print(s)

- rev = ''.join(reversed(s))

# Checking if both string are equal or not

- if (s == rev):
  - print("The string is a palindrome.")
- else:
  - print("The string is not a palindrome.")

03.

cycle through each character in the provided string using a for loop joining it with each element kept in an empty variable we define.

```
#Enter input string
    string = input("Enter string : ")
#Declare an empty string variable
    revstr = ""
#Iterate string with for loop
    for i in string:
        revstr = i + revstr
        print("Reversed string : ", revstr)
        if(string == revstr):
            print("The string is a palindrome.")
        else:
            print("The string is not a palindrome.")
```

04. Using recursion

```
def isPalindrome(string):
    if len(string)<1:
        return True
    else:
        if string[0]==string[-1]:
            return isPalindrome(string[1:-1])
        else:
            return False
str1=input("Enter string:")
if (isPalindrome(str1)==True):
print(" The string is a palindrome.")
else:
print("The string is not a palindrome.")
```