ES2B4 Computer Engineering and Programming: Assignment

SCHOOL OF ENGINEERING – UNIVERSITY OF WARWICK

NAME: PABLO NAVAJAS HELGUERO

ID: 1604648

SECTION: A1. Basic Operators and Order of Execution

A1.1 Question: Are there any common traits?
Why are results so different?

The operations in this part of the Assignment show several common traits. First of all, it can be seen that we can operate with both floats (fractional or decimal numbers) and Integers (whole number) and combine them. We can also determine the operators for certain numeric operations, such as the use of '*' to multiply and '/' to divide. In addition, the '//' operator: also called integer division, which retains only the whole part of the result, eliminating the decimals.

Despite using the same operators, the results vary significantly depending on the form of the equation. This is due to the order of execution. It can be inferred from the results that Python automatically follows the convention for order of execution, which is: Brackets, Order of Power, Division and Multiplication, and Addition and Subtraction. Thus, the addition of brackets can change drastically the result of an operation (as shown by the 7th and 8th Entry in Image1).

```
In [1]: 5/2
Out[1]: 2.5

In [2]: 5/2.0
Out[2]: 2.5

In [3]: 5.0/2
Out[3]: 2.5

In [4]: 5//2
Out[4]: 2

In [5]: 5.0//2.0
Out[5]: 2.0

In [6]: 7*(1/2)
Out[6]: 3.5

In [7]: 7*(1//2.0)
Out[7]: 0.0

In [8]: 7*1//2.0
Out[8]: 3.0

In [9]: 1/3.0
Out[9]: 0.3333333333333333

In [10]: 5**2.0
Out[10]: 25.0

In [11]: 5.0**2.0
Out[11]: 25.0

In [12]: 5**2
Out[12]: 25
```

Image 1: Record operations A1.1

A1.2 & 3 Exercises attached as Files:
assignment_a1_2.py and assignment_a1_3.py

These sections show the importance of the order of execution when transforming an expression into Python. In order to achieve the desired outcome it is essential to check the order of the operations and use parentheses when necessary.

*The equation used for part A1.3 is:

$$\frac{\sqrt[3]{11-3}\cdot\left(\frac{4^2+2}{3^2}+\frac{3^2+2}{11}\right)}{\frac{6\cdot 3^2}{2}-\sqrt{64}\cdot 3+3}=1 \qquad (1)$$

A1.4 Other Operations: Explain your observations

```
In [1]: 'abc'>123
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-1-5ee93968e571> in <module>()
----> 1 'abc'>123

TypeError: '>' not supported between instances of 'str' and 'int'
```

Image 2: Record Expression 1 A1.4

From the first expression's error it can be inferred that it is not possible to compare strings and integers.

The second operation highlights the fact that strings can be compared between them as if they were numbers. This is due to the fact that each character in a computer is represented as a numeric value, following the American Standard Code for Information Interchange (ASCII: shown in Table 1). Each character has a value based on the lexicographical order. In this case, 'apple' has a lower value than 'banana' due to the fact that the combination of values of the letters in 'banana' is higher.

```
In [2]: 'apple'<'banana'
Out[2]: True

In [3]: 'oranges'<'ORANGES'
Out[3]: False

In [4]: a=True;

In [5]: b=(5>7)

In [6]: a == b
Out[6]: False

In [7]: str = ""

In [8]: not str
Out[8]: True

In [9]: string = "0"

In [10]: not string
Out[10]: False
```

Image 3: Expressions 2 to 5 A1.4

The third expression shows that the letter case affects their ASCII value. Uppercase letters have a lower value (from 65 to 90) than lowercase letters (from 97 to 122).

The fourth comparison introduces Boolean logic. It can be seen that numeric values can be evaluated as Boolean expressions returning a true or false outcome. Given that the expression in 'b' (5>7) is false and 'a' is true, when checking if they are equal Python indicates 'False': they are not.

Finally, the fifth and sixth operation illustrate the fact that strings can also be evaluated as Boolean expressions, and introduce the Boolean operation: 'not'. This operator inverts the form of checking if a string is true (has content) or false (it's empty). This explains why the output of the fifth exercise is True (the string is empty meaning it should be 'False' but it is inverted, turning it into 'True'); and, following the same logic, why the result of the sixth exercise is False (It holds the value "0" and; therefore, it is not empty. It would have been empty if it had been 'string = 0').

Table 1: ASCII TABLE

Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char
0	0	0		32	20	40	[space]	64	40	100	@	96	60	140	`
1	1	1		33	21	41	!	65	41	101	A	97	61	141	a
2	2	2		34	22	42	"	66	42	102	B	98	62	142	b
3	3	3		35	23	43	#	67	43	103	C	99	63	143	c
4	4	4		36	24	44	$	68	44	104	D	100	64	144	d
5	5	5		37	25	45	%	69	45	105	E	101	65	145	e
6	6	6		38	26	46	&	70	46	106	F	102	66	146	f
7	7	7		39	27	47	'	71	47	107	G	103	67	147	g
8	8	10		40	28	50	(	72	48	110	H	104	68	150	h
9	9	11		41	29	51	)	73	49	111	I	105	69	151	i
10	A	12		42	2A	52	*	74	4A	112	J	106	6A	152	j
11	B	13		43	2B	53	+	75	4B	113	K	107	6B	153	k
12	C	14		44	2C	54	,	76	4C	114	L	108	6C	154	l
13	D	15		45	2D	55	-	77	4D	115	M	109	6D	155	m
14	E	16		46	2E	56	.	78	4E	116	N	110	6E	156	n
15	F	17		47	2F	57	/	79	4F	117	O	111	6F	157	o
16	10	20		48	30	60	0	80	50	120	P	112	70	160	p
17	11	21		49	31	61	1	81	51	121	Q	113	71	161	q
18	12	22		50	32	62	2	82	52	122	R	114	72	162	r
19	13	23		51	33	63	3	83	53	123	S	115	73	163	s
20	14	24		52	34	64	4	84	54	124	T	116	74	164	t
21	15	25		53	35	65	5	85	55	125	U	117	75	165	u
22	16	26		54	36	66	6	86	56	126	V	118	76	166	v
23	17	27		55	37	67	7	87	57	127	W	119	77	167	w
24	18	30		56	38	70	8	88	58	130	X	120	78	170	x
25	19	31		57	39	71	9	89	59	131	Y	121	79	171	y
26	1A	32		58	3A	72	:	90	5A	132	Z	122	7A	172	z
27	1B	33		59	3B	73	;	91	5B	133	[	123	7B	173	{
28	1C	34		60	3C	74	<	92	5C	134	\	124	7C	174	|
29	1D	35		61	3D	75	=	93	5D	135	]	125	7D	175	}
30	1E	36		62	3E	76	>	94	5E	136	^	126	7E	176	~
31	1F	37		63	3F	77	?	95	5F	137	_	127	7F	177	

SECTION: A2. Simple Control Flow

Exercise attached as File: assignment_a2.py

This exercise covers the conditional execution statements: 'if', 'elif' and 'else'. It works with them by creating a model of the Rock-Paper-Scissors game. Using a Truth Table (Table 2), we can see the outcome of the different inputs to write our code for a 2-player game.

Table 2: Truth Table A2

PLAYER 1	PLAYER 2	WINNER
Rock	Rock	Tie
Rock	Paper	Player 2
Rock	Scissors	Player 1
Paper	Paper	Tie
Paper	Scissors	Player 2
Paper	Rock	Player 1
Scissors	Scissors	Tie
Scissors	Rock	Player 2
Scissors	Paper	Player 1

The use of 'nested if' not only enables the program to determine the appropriate outcome of the game according to the input of the players, but it also identifies the player who has introduced a wrong choice. Considering that it was not specified to be an infinite loop, the program will only run once each time.

```
In [1]: runfile('/Users/pablonavajas/Documents/Assignment_a2.py', wdir='/Users/pablonavajas/
Documents')

Player 1! Do you choose Rock, Paper or Scissors? : Paper

And Player 2! Do you choose Rock, Paper or Scissors? : Paper
It's a Tie!

In [2]: runfile('/Users/pablonavajas/Documents/Assignment_a2.py', wdir='/Users/pablonavajas/
Documents')

Player 1! Do you choose Rock, Paper or Scissors? : Scissors

And Player 2! Do you choose Rock, Paper or Scissors? : Rock
Player 2 wins..!!!

In [3]: runfile('/Users/pablonavajas/Documents/Assignment_a2.py', wdir='/Users/pablonavajas/
Documents')

Player 1! Do you choose Rock, Paper or Scissors? : Rock

And Player 2! Do you choose Rock, Paper or Scissors? : Rockk
Player 2, you have entered a wrong choice
```

Image 4: Sample Run of the 'Rock-Paper-Scissors' Game A2

SECTION: A3. Loops

A3.1, 2 & 3 Exercises attached as Files: assignment_a3_1.py, assignment_a3_2.py and assignment_a3_3.py

These sections are focused on creating loops using both 'while' and 'for' statements. As it was not specified, none is an intentionally infinite loop: the program runs once for each entry.

A3.1 - For the first sub-section, I created a program that runs in a loop checking if the input is a palindrome. A palindrome is a sequence of characters that reads the same backward and forward. In order to detect the palindrome regardless of the special characters used, I introduced the 'casefold' option to ignore the letter case and the 'maketrans' method to overlook the punctuation and spaces. This code allows the program to check if words, sentences and numbers are Palindromes (shown in Image 5).

```
In [1]: runfile('/Users/pablonavajas/Documents/Assignment_a3_1.py', wdir='/Users/pablonavajas/
Documents')

Introduce your Palindrome: Madam
Yes.  Madam  is a Palindrome!

In [2]: runfile('/Users/pablonavajas/Documents/Assignment_a3_1.py', wdir='/Users/pablonavajas/
Documents')

Introduce your Palindrome: Kayaks
Sorry.  Kayaks  is not a Palindrome!

In [3]: runfile('/Users/pablonavajas/Documents/Assignment_a3_1.py', wdir='/Users/pablonavajas/
Documents')

Introduce your Palindrome: A man, a plan, a canal, Panama!
Yes.  A man, a plan, a canal, Panama!  is a Palindrome!

In [4]: runfile('/Users/pablonavajas/Documents/Assignment_a3_1.py', wdir='/Users/pablonavajas/
Documents')

Introduce your Palindrome: 1440441
Yes.  1440441  is a Palindrome!
```

Image 5: Palindrome Checker A3.1

A3.2 - For the next exercise, we had to create a 'count down timer' using a while loop. To ensure that it would work as a timer I imported the 'time' library in Python. After asking the user for the input, the program transforms it into an integer. If it can't be transformed (the value entered is not a number): it prints a message informing the user that the input needs to be a valid number.

Depending on the transformed integer, the program follows two different procedures. If it is a negative number, it states that it is not a valid input (this is due to the fact that time is not measured in negative numbers and the program would be counting down indefinitely). Finally if the Input is a positive number, it performs the countdown leaving a one second gap between each number (thanks to the 'time.sleep' functionality); and, once it reaches zero, it prints: 'Finish!'.

```
In [1]: runfile('/Users/pablonavajas/Documents/Assignment_a3_2.py', wdir='/Users/pablonavajas/
Documents')

Introduce starting value: 6
6
5
4
3
2
1
Finish!

In [2]: runfile('/Users/pablonavajas/Documents/Assignment_a3_2.py', wdir='/Users/pablonavajas/
Documents')

Introduce starting value: -5
That's not a valid input.

In [3]: runfile('/Users/pablonavajas/Documents/Assignment_a3_2.py', wdir='/Users/pablonavajas/
Documents')

Introduce starting value: Eight
You need to introduce a valid number.
```

Image 6: Count Down Timer A3.2

A3.3 – The last sub-section involved building a program that could calculate the factorial of a number. As in the previous exercise, the program converts the input into an integer in search of invalid entries; and, in case it is not a number, it shows a message informing the user that it should be a number. After conversion, it checks if it is a negative number, in which case it explains that factorials of negative numbers do not exist. Finally, if it is a positive number, it calculates the result using a 'for' loop and prints the result of the factorial (as seen in Image 7). The equation used to calculate the factorial is:

$$Factorial = (Factorial \cdot i) \;\; for\, i \in (1 \ldots Input) \qquad (2)$$

*As 'Factorial' is also used in the equation the desired value is updated in loop until it reaches the value of the Input number (Factorial =1·1=…= n-1! · n = n!).

```
In [1]: runfile('/Users/pablonavajas/Documents/Assignment_a3.3.py', wdir='/Users/pablonavajas/
Documents')

Introduce number: 12
The factorial of 12  is  479001600

In [2]: runfile('/Users/pablonavajas/Documents/Assignment_a3.3.py', wdir='/Users/pablonavajas/
Documents')

Introduce number: 5
The factorial of 5  is  120

In [3]: runfile('/Users/pablonavajas/Documents/Assignment_a3.3.py', wdir='/Users/pablonavajas/
Documents')

Introduce number: -9
Sorry, factorials of negative numbers do not exist.

In [4]: runfile('/Users/pablonavajas/Documents/Assignment_a3.3.py', wdir='/Users/pablonavajas/
Documents')

Introduce number: 0
The factorial of 0 is 1.

In [5]: runfile('/Users/pablonavajas/Documents/Assignment_a3.3.py', wdir='/Users/pablonavajas/
Documents')

Introduce number: P
That is not a NUMBER. You need to enter a valid number.
```

Image 7: Factorial of a Number A3.3

SECTION: A4. Cyclic Cipher

This task required creating a Cyclic Cipher: an apparatus to code messages developed by Julius Caesar. The concept consists of two concentric circular alphabets in which the inner circle can rotate creating a new alignment of letters. This shift shows how letters are coded, in other words: how has the message been encrypted (shown in Image 8).

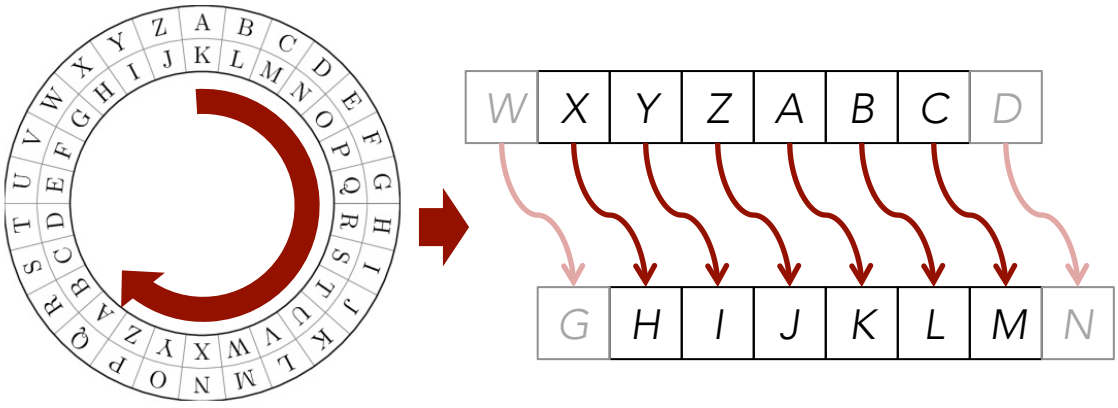

Image 8: Cyclic Cipher Schematic

The exercise involved working with loops and conditionals to ensure that the message introduced in the program was successfully coded. Considering that special characters had to be ignored, I decided to tackle it in four different parts: one for each set of characters that needed to be considered: Numbers, Uppercase letters, Lowercase letters and Special Characters.

The program runs in an intentional infinite loop thanks to the use of Boolean logic. To close the program, I established that the user should enter the message: "Exit cipher", which was achieved through an 'if' condition. Additionally, the program converts the Shift value into an integer and checks if it is a valid or an invalid input (a number or not). If it can't be converted, the program will state that it is not a valid Shift value and will restart the loop. It will not terminate the program thanks to a 'continue' statement.

To begin with the encryption, I translated the characters from the Input message into their ASCII values. Consequently, I established the conditions to divide them in Numbers, Uppercase letters, Lowercase letters and Special characters. Once they are all allocated to their respective range, the program follows two different procedures:

For Numbers, Uppercase letters and Lowercase letters:

The program checks if the Input for the Shift value is higher than the range available. In other words, the system checks if adding the shift to the ASCII values will take them out of their range. If this is the case, it adjusts it to make it lower through this equation:

$$NewShift = Shift - [Range \cdot (Shift//Range)] \quad (3)$$

*This equation uses the integer division to subtract the number of complete laps around the cipher from the original shift (the whole part of the division equals how many times has the inner circle of the cipher completed a spin).

Once the shift is adjusted, the program adds it to the ASCII value and checks that the encrypted character is under the maximum value of the range. If the encrypted character has exceeded the Maximum value: it will deduct from them the number of elements in its range (10 in the case of Numbers and 26 for both Upper and Lower Case letters). This ensures that the shifted values will stay within their corresponding range: when the last value is reached, it will return to the first one. After the ASCII values have been shifted, they are each transformed into their respective encrypted character.

For Special Characters:

Special characters should be ignored so they do not undergo the previous process. They are simple transformed back into characters.

Finally, the Encrypted Characters from both processes are printed in the same line, and the loop runs again.

```
In [1]: runfile('/Users/pablonavajas/Documents/assignment_A4.py', wdir='/Users/pablonavajas/Documents')

Introduce your Message: I am 19 Years Old...

Introduce the shift value: 7

Your encrypted message is:
P ht 89 Flhyz Vsk...

Introduce your Message: It WoRkS!!!

Introduce the shift value: 43

Your encrypted message is:
Zk NfIbJ!!!

Introduce your Message: Checking invalid SHIFT:

Introduce the shift value: P
That is not a valid shift. Enter a number!

Introduce your Message: 100% PrEcIsIoN - perfect

Introduce the shift value: -38

Your encrypted message is:
322% DfSqWgWcB - dsftsqh

Introduce your Message: Exit cipher

Introduce the shift value: (It doesn't matter what I write now)
Exiting from the program!
```

Image 9: Cyclic Cipher Sample A4