LC CS Python

Student Exercise Book

LEAVING CERTIFICATE
COMPUTER SCIENCE

Section 3

Strings

Name: _________________________________

Task 1

Experiment!

A **pangram** is a sentence that uses every letter of the alphabet at least once. Study the program below and see if you can predict what it does? Record your prediction on the right hand side.

Key in the program and make some changes? What happens?

```
1.  # Program to demonstrate basic string operations
2.
3.  # Initialise the string
4.  pangram = "The quick brown fox jumps over the lazy dog!"
5.  #          01234567890123456789012345678901234567890123
6.  # INDEXING
7.  print(pangram[0])
8.  print(pangram[1])
9.  print(pangram[2+4])
10. print(pangram[14])
11. print(pangram[8])
12. print(pangram[43])
13.
14. # The index can be any valid Python expression
15. pos = 17
16. print(pangram[pos])
17. print(pangram[pos+1])
18.
19. # A general pattern used to find the last character
20. print( pangram[len(pangram)-1] )
```

Prediction

Line:

7.

8.

9.

10.

11.

12.

16.

17.

20.

Task 2

Read through the following program. Do you understand what each line of code does?

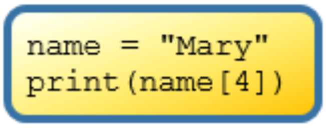
```
# Program to demonstrate basic string slicing

# Initialise the string
pangram = "The quick brown fox jumps over the lazy dog!"
#          01234567890123456789012345678901234567890123

# Extract from index 1 up to but NOT including index 5
print(pangram[1:5]) # "he q"

# Extract from index 17 up to but NOT including index 19
print(pangram[17:19]) # ox

print(pangram[:19])   # "The quick brown fox"
print(pangram[20:26]) # "jumps"
print(pangram[26:])   # "over the lazy dog!"
```

What do you think the statement `print(pangram[:])` would display?

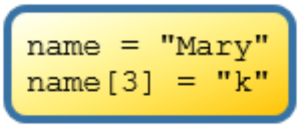
Task 3

Q1. Explain what is wrong with each of the following code snippet.

Q2. What do you think the programmer was trying to achieve in the second example?



```
name = "Mary"
print(name[4])
```



```
name = "Mary"
name[3] = "k"
```

____

____

____

____

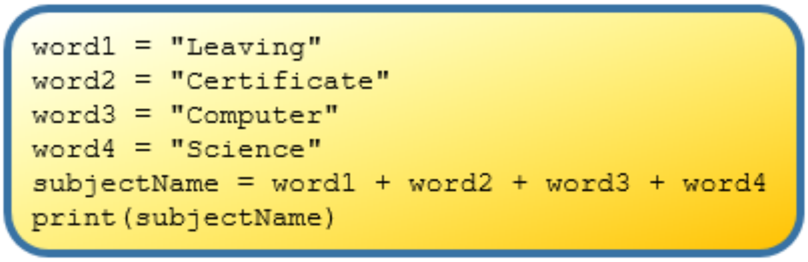
____

Task 4

These programs below illustrate how the plus operator is used to **concatenate** strings.

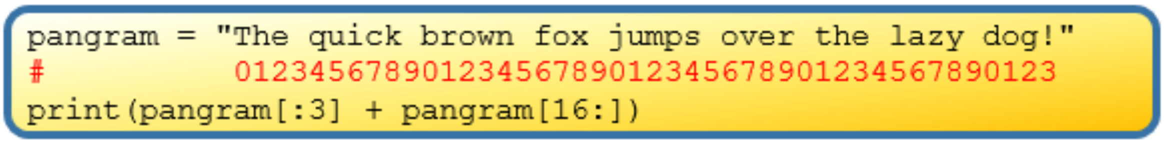
- Can you predict the output for each program? Make your predictions on your whiteboard.
- Type in each program and get them to run. Did the *actual outputs* match your *predicted outputs*?

PROGRAM 1



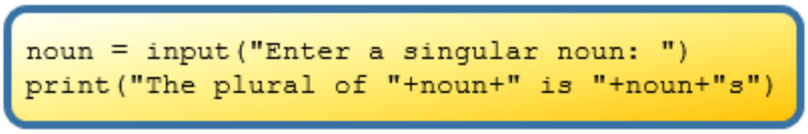
```
word1 = "Leaving"
word2 = "Certificate"
word3 = "Computer"
word4 = "Science"
subjectName = word1 + word2 + word3 + word4
print(subjectName)
```

PROGRAM 2



```
pangram = "The quick brown fox jumps over the lazy dog!"
#          01234567890123456789012345678901234567890123
print(pangram[:3] + pangram[16:])
```

PROGRAM 3



```
noun = input("Enter a singular noun: ")
print("The plural of "+noun+" is "+noun+"s")
```

Log your thoughts.
Each of the above programs contains a deliberate subtle error.
Can you suggest any improvements?

Task 5

The string variable `alNum` is initialised as shown here.

```
alNum = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789"
#                  1         2         3         4         5         6
#        01234567890123456789012345678901234567890123456789012345678901
```

Match each index operation on `alNum` to the correct value in the middle

`alNum[3]`	ABCDEFGHIJKLMNOPQRSTUVWXYZ0
`alNum[52:62]`	d
`alNum[51:62]`	ABCDEFGHIJKLMNOPQRSTUVWXYZ
`alNum[26:53]`	A
`alNum[0:25]`	0123456789
`alNum[1]`	f
`alNum[5]`	b
`alNum[26:52]`	abcdefghijklmnopqrstuvwxyz
`alNum[26]`	Z0123456789

1	2	3	4	5	6	7	8	9

Task 6

The four statements below each generate the same output.
What output is displayed?

```
print("2 + %d = 4"   %2)
print("2 + %d = %d"  %(2, 4))
print("%d + %d = %d" %(2, 2, 4))
print("%d + %d = %d" %(2, 2, 2+2))
```

Task 7

Experiment!
See if you can figure out what the following code does.

```
print("%s" %3)
print("%d" %3.14)
print("%f" %3)
print("%f" %"Hi!")
```

Task 8

Experiment!
See if you can figure out what the following code does.

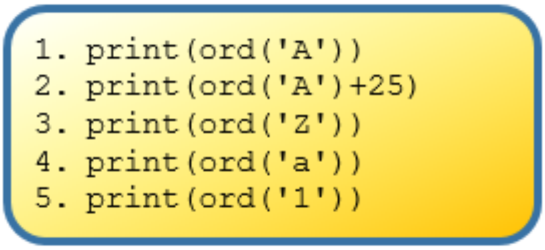
```
msg = "Hi %s. How are you?"
name = "Hal"
print(msg%name)
```

```
import math
r = 5
print("Radius: %d, Area: %.2f" %(r, 2*math.pi*r))
```

The second snippet of code contains a semantic error. Can you identify it?

Task 9

The programs below illustrate the use of `ord`.



```
1. print(ord('A'))
2. print(ord('A')+25)
3. print(ord('Z'))
4. print(ord('a'))
5. print(ord('1'))
```

Line 1 outputs 65. This is the ASCII representation for the character ‘A’

Line 2 outputs 90, the ASCII representation for the character ‘Z’ (25 letters from ‘A’

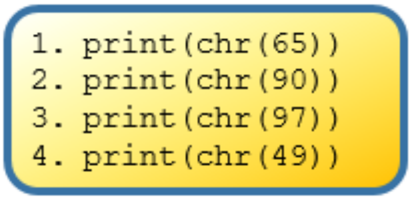
Use the ASCII table to predict the outputs of lines 3, 4 and 5 of the above program.

Line 3:

Line 4:

Line 5:

The programs below illustrate the use of `chr`. Fill in the blanks!



```
1. print(chr(65))
2. print(chr(90))
3. print(chr(97))
4. print(chr(49))
```

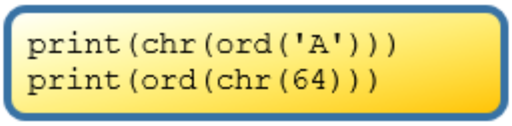
Line 1 outputs character ‘A’

Line 2 outputs character ‘Z’

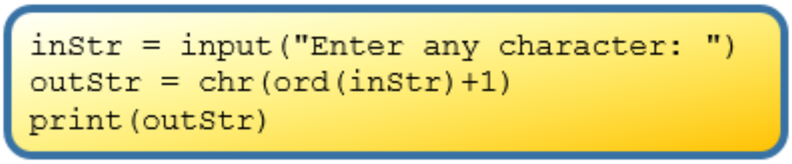
Line 3 outputs… ______________________________

Line 4 outputs… ______________________________

Experiment!
Look at the following programs and see if you can explain what is going on.



```
print(chr(ord('A')))
print(ord(chr(64)))
```



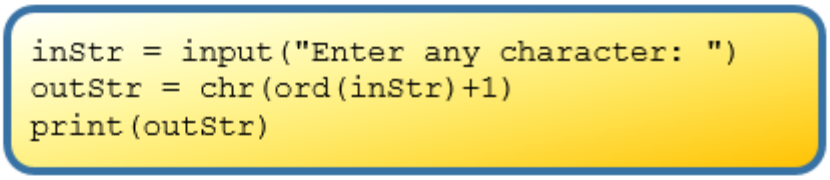
```
inStr = input("Enter any character: ")
outStr = chr(ord(inStr)+1)
print(outStr)
```

Task 10

Programming Exercise

A Caesar cipher is a way of encoding strings by shifting each letter by a fixed number of positions (called a key) in the alphabet. For example, if the key value is 1, the string ’LCCS’. would be encoded as ‘MDDT’. Each letter of LCCS is moved on by one.

The short program below prompts a user to enter a single character and then it calculated and displays the character with the next ordinal number e.g. if the user enters A the program will display B.



```
inStr = input("Enter any character: ")
outStr = chr(ord(inStr)+1)
print(outStr)
```

Type in the program and test it. Once you understand what the program does change it so that it "encodes" a 6 letter string using a key value of 1 e.g. "Python" becomes "Qzuipo".

In order to complete this task, you will need to understand:

- ✓ Variables and assignments
- ✓ Strings (indexing and concatenation)
- ✓ Data representation (ASCII/Unicode)
- ✓ How to use the `ord`, `chr` and `print` built-in functions

Task 11

The code below initialises four string variables

```
# Initialise four string variables
lowerLetters = "abcdefghijklmnopqrstuvwxyz"
upperLetters = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
allLetters = lowerLetters+upperLetters
digits = "0123456789"
```

Match each Python statement to the correct value shown on the right

```
print(lowerLetters.upper())
print(upperLetters.lower())
print(upperLetters.find("h"))
print(lowerLetters.find("h"))
print(digits.count("123"))
print(lowerLetters.capitalize())
print(digits.count("0"))
print(digits.lower())
print( min(allLetters) )
print( max(allLetters) )
print(allLetters.count("a"))
print(allLetters.count("aA"))
print(allLetters.count("ABC"))
print(digits.replace("0", "1"))
print(allLetters.replace("ABC", "ABC"))
print(allLetters.replace("xyz", "ABC"))
```

```
a
Z
abcdefghijklmnopqrstuvwxyz
0123456789
ABCDEFGHIJKLMNOPQRSTUVWXYZ
-1
1
z
A
7
0
abcdefghijklmnopqrstuvwABCABCDEFGHIJKLMNOPQRSTUVWXYZ
Abcdefghijklmnopqrstuvwxyz
abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ
6
1123456789
```

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16