LC CS Python

Student Exercise Book

LEAVING CERTIFICATE
COMPUTER SCIENCE

Section 6

Functions

Name: _____

Task 1 – Maximising code reuse and minimising redundancy

Read through the following code and try to predict the output to screen. Type up the code and run it. Were your predictions correct?

```
# A program to display Green Eggs and Ham
1. print("I do not like green eggs and ham.")
2. print("I do not like them Sam-I-am.")
3. print() # display a blank line
4. print("I do not like them here or there.")
5. print("I do not like them anywhere.")
6. print("I do not like them in a house")
7. print("I do not like them with  a mouse")
8. print() # display a blank line
9. print("I do not like green eggs and ham.")
10.print("I do not like them Sam-I-am.")
11.print() # display a blank line
12.print("I do not like them in a box")
13.print("I do not like them with a fox")
14.print("I will not eat them in the rain.")
15.print("I will not eat them on a train")
16.print() # display a blank line
17.print("I do not like green eggs and ham.")
18.print("I do not like them Spam-I-am.")
```

Evaluate the above program by asking - do you recognise any patterns? Is there any code duplication? Can you spot any typing mistakes? Elaborate.

Task 2 – using a function to eliminate duplication of code

Read through the following code and try to predict the output to screen. Type up the code and run it. Were your predictions correct?

```
# A program to display Green Eggs and Ham (v2)
1.  def displayChorus():
2.      print()
3.      print("I do not like green eggs and ham.")
4.      print("I do not like them Sam-I-am.")
5.      print()
6.
7.  displayChorus()
8.  print("I do not like them here or there.")
9.  print("I do not like them anywhere.")
10. print("I do not like them in a house")
11. print("I do not like them with  a mouse")
12. displayChorus()
13. print("I do not like them in a box")
14. print("I do not like them with a fox")
15. print("I will not eat them in the rain.")
16. print("I will not eat them on a train")
17. displayChorus()
```

Task 3 – Using *functions* to write code which is more modular

Read through the following code and try to predict the output to screen. Type up the code and run it. Were your predictions correct?

```
# A program to display Green Eggs and Ham (v3)
1.  def displayChorus():
2.      print()
3.      print("I do not like green eggs and ham.")
4.      print("I do not like them Sam-I-am.")
5.      print()
6.
7.  def displayVerse1():
8.      print("I do not like them here or there.")
9.      print("I do not like them anywhere.")
10.     print("I do not like them in a house")
11.     print("I do not like them with  a mouse")
12.
13.  def displayVerse2():
14.     print("I do not like them in a box")
15.     print("I do not like them with a fox")
16.     print("I will not eat them in the rain.")
17.     print("I will not eat them on a train")
18.
19.  displayChorus()
20.  displayVerse1()
21.  displayChorus()
22.  displayVerse2()
23.  displayChorus()
```

Task 4

Reflect on what you have learned about functions so far. Use the space below to explain what functions are and why they are important.

Task 5

Use functions to display your favourite song (hint: verse 1, chorus, verse 2, chorus, etc)

Task 6

```
1. def homework(): # function header
2.     print("Jack loves to do his homework")
3.     print("He never misses a day")
4.     print("He even loves the men in white")
5.     print("Who are taking him away")
6.     # End of function
7.
8. homework() # call the function
9. print("The End!")
```

Describe how the above program could be modified so that it would output the text shown below. Type up your solution and verify that it works.

Can you come up with more than one solution and if so which solution do you think is better?

The Start!
Jack loves to do his homework
He never misses a day
He even loves the men in white
Who are taking him away
The End!

Task 7

Design and write a function to display the output shown below.

Write a line of code to call your function.

Way down south where banannas grow,
A grasshopper stepped on an elephant's toe.
The elephant said, with tears in its eyes,
'Pick on somebody your own size.'

Task 8

Study the program shown carefully and identify any error(s).

```
# A program to display Green Eggs and Ham (v3)
1. def showChorus():
2.     print()
3.     print("I do not like green eggs and ham.")
4.     print("I do not like them Sam-I-am.")
5.     print()
6.
7. def showVerse1():
8.     print("I do not like them here or there.")
9.     print("I do not like them anywhere.")
10.    print("I do not like them in a house")
11.    print("I do not like them with  a mouse")
12.
13. def showVerse2():
14.    print("I do not like them in a box")
15.    print("I do not like them with a fox")
16.    print("I will not eat them in the rain.")
17.    print("I will not eat them on a train")
18.
19. displayChorus()
20. displayVerse1()
21. displayChorus()
22. displayVerse2()
23. displayChorus()
```

SYNTAX CHECK: When Python comes across a word it does not understand it displays a syntax error. Therefore, when a call is made to a function using a name, a function of that name must already have been made known to Python with the `def` keyword.

Task 9

Comment on the validity and quality of each of the following function names

a) sendTweet
b) calculate_salary
c) _login
d) Bin
e) 2binary
f) MAX_OF_3
g) Binary Search
h) Search*
i) Return
j) Print
k) doSomething

Task 10

a. Type up the code below and run it.

```
1. def homework(): # function header
2.     print("Jack loves to do his homework")
3.     print("He never misses a day")
4.     print("He even loves the men in white")
5.     print("Who are taking him away")
6.     # End of function
7.
8. homework() # call the function
9. print("The End!")
```

Every time this function is called it always displays precisely the same text. We'd like to move away from the concrete name – Jack – to a more abstract name – anybody.

b. Modify the code so that the function takes a parameter called *personName* and prints the contents of this parameter in the first line of the poem.

```
1. def homework(personName): # function header
2.     print(personName, "loves to do his homework")
3.     print("He never misses a day")
4.     print("He even loves the men in white")
5.     print("Who are taking him away")
6.     # End of function
7.
8. homework("David") # call the function
```

David loves to do his homework
He never misses a day
He even loves the men in white
Who are taking him away

c. Add lines to the code which call the *homework* function with name other than David.

Would it make sense to use a girl's name here?

Task 11

Type up the code below and run it.

```
# Functions can have multiple parameters
1. def displayMessage(msg1, msg2):
2.     print(msg1)
3.     print(msg2)
4.     # End of function
5.
6. displayMessage("Hello world", "How are you today?")
```

Hello World
How are you today?

a. Identify one **parameter** and one **argument** in the code above.

Parameter: _____

Argument: _____

b. Modify line 6 to the line given below. Are you surprised by the output?

```
6.  displayMessage("How are you today?", "Hello world")
```

Task 12

Predict what the following calls to the function `displayMessage` would do? After making each prediction you should use Python to see if you were correct.

1. `displayMessage(1, 2)`

_____

_____

2. `displayMessage(1, 2, 3)`

_____

_____

3.
```
displayMessage("Testing", 123)
```

4.
```
displayMessage("Testing", 1 2 3)
```

5.
```
str1 = "Hello world"
str2 = "How are you today?"
displayMessage(str1, str2)
```

6.
```
str1 = "Hello world"
str2 = "How are you today?"
displayMessage(str2, str1)
```

7.
```
str1 = "Dear "
str2 = "John"
displayMessage(str1+str2, "Hi!")
```

8.
```
displayMessage(1+1, 2)
```

9.
```
pi = 3.14
r = 5
displayMessage("Area:", pi*r**2)
```

Task 13

Reflect.

What conclusion(s) about the syntax and semantics of parameters and arguments did you arrive at from the previous experiment?

Task 14

```
# Add all the numbers from 0 to n
1. def sumOfN(n):
2.     total = 0
3.
4.     for i in range(n+1):
5.         total = total + i
6.
7.     return total
```

Experiment!
Key in the above code and make the function call.
Explain why nothing appears to happen?

Task 15

Modify the code so that the function is called, the return value is assigned to the variable *answer* and the result is printed to screen.

```
# Add all the numbers from 0 to n
1.  def sumOfN(n):
2.      total = 0
3.
4.      for i in range(n+1):
5.          total = total + i
6.
7.      return total
8.
9.  answer = sumOfN(10) # call the function and save the result
10. print("The sum of the first 10 integers is %d" %answer)
```

Task 16

Modify the code so that the function is called within the *print* statement.

```
# Add all the numbers from 0 to n
1.  def sumOfN(n):
2.      total = 0
3.
4.      for i in range(n+1):
5.          total = total + i
6.
7.      return total
8.
9.  # call the function and print the result
10. print("The sum of the first 10 integers is", sumOfN(10))
```

Task 17

Type up and run the following code. Can you explain why the program prints 3, -5, and -2?

```
def sub(a, b):
    answer = a - b
    return answer

print( sub( 2, sub(3, 4)))
print( sub( sub(2, 3), 4))
print( sub( 2, sub(3, sub(4, 5))))
```

Task 18 – Temperature and Distance Conversions

The program below implements two functions:

(i) `cent2fhar` converts from Centigrade to Fahrenheit and

(ii) `kms2miles` converts from kilometres to miles

```
# Convert centigrade to fharenheit
def cent2fhar(centigrade):
    return 9/5*centigrade + 32

# Convert kilometers to miles
def kms2miles(kilometers):
    return( kilometers * 0.62)


# Test cent2fhar
cent = int(input("Enter a value in centigrade: "))
fhar = cent2fhar(cent)
print(cent, "degrees C is", fhar, "degrees F")

# Test kilometers to miles conversion
kms = int(input("Enter a value in kilometers: "))
print("%dkms = %.2fmiles" %(kms, kms2miles(kms)))
```

What you have learned about functions from this example?

Extend the program with functions to convert from a) miles to kilometres and b) Fahrenheit to Centigrade

Task 19 – Compound Interest

The program below can be used to find out which would yield a greater future value:

Scenario A: €10,000 at a rate of 5% for 10 years or

Scenario B: €10,000 at a rate of 10% for 5 years

```
# Calculates future value
def future_value(principal, rate, time):
    fv = principal * (pow((1 + rate / 100), time))
    return fv

# Driver Code
fv_A = future_value(10000, 5, 10)
fv_B = future_value(10000, 10, 5)
print("Scenario A: %.2f \nScenario B: %.2f" %(fv_A, fv_B))
```

The formula used is $\boldsymbol{FV = p(1+i)^t}$ where, p is the initial principal, i the interest rate, expresses as a decimal, and t the time in years.

(`pow(x,y)` is a built-in function that returns `x` raised to the power of `y`.)

What you have learned about passing information in and out of functions from this example?

1. Modify the program so that it can accept the values for principle, interest and time from the end-user
2. Extend the program so that it can compare compound interest with simple interest

STUDENT TIP

When you want to pass information out of a function use `return`

Task 20 – Maximum of Three Numbers

The function below finds and returns the largest of any three integers.

The solution is based on three, two-way comparisons in which each of the integers is compared to the other two. The individual comparisons are combined into a compound Boolean condition using the `and` operator.

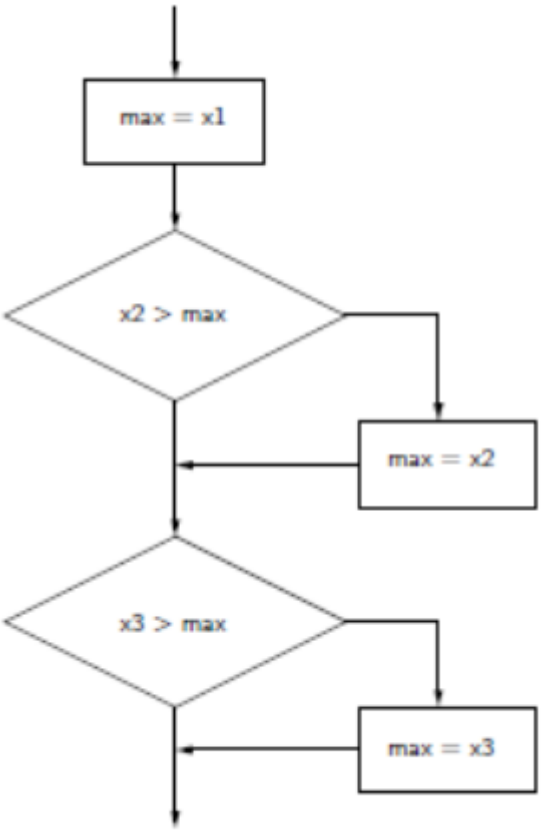
```
# A function to find the largest of 3 numbers
def maxOf3(x, y, z):
    if (x > y) and (x > z):
        return x
    elif (y > x) and (y > z):
        return y
    elif (z > x) and (z > y):
        return z

# Test the function
print(maxOf3(1, 2, 3))
print(maxOf3(1, 3, 2))
print(maxOf3(3, 2, 1))
```

Reflect.

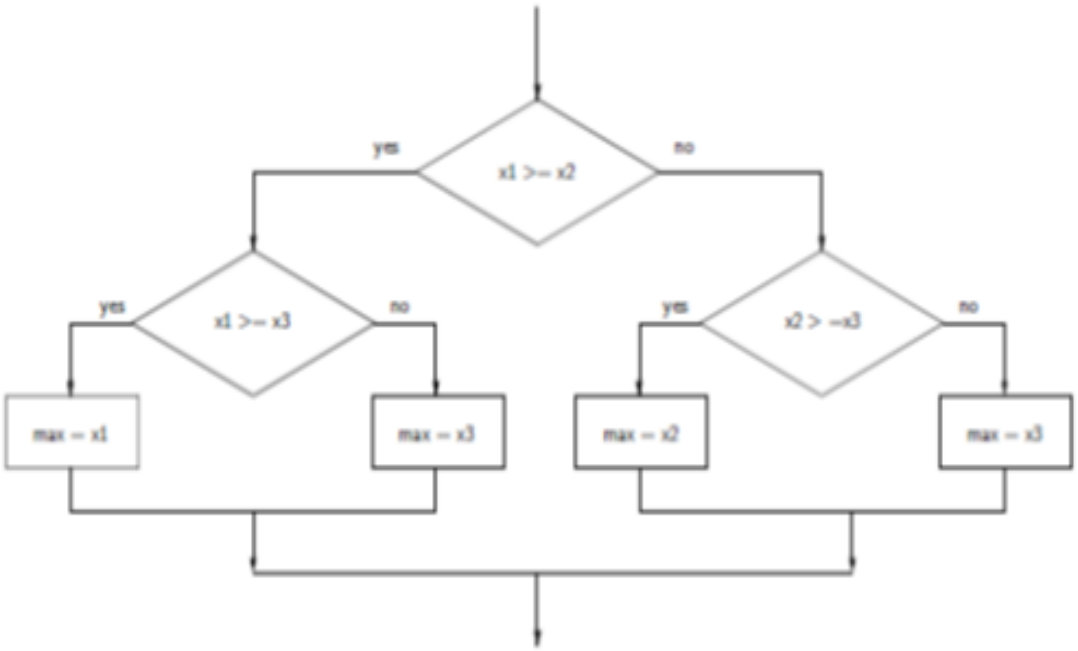
How has this example extended your knowledge in relation to forming Python Boolean expressions?

1. Modify the program so that it prompts the end-user to enter three values and then displays their maximum in a meaningful output message.
   How would this implementation be altered if the three numbers were a) randomly generated or b) read from a file?
2. Use the flowcharts on the next page to implement two alternative solutions for `maxOf3`
3. Design a flowchart to find the minimum of three numbers.
4. Use your flowchart to implement a function to find the minimum of three values. Call it `minOf3`. You could start by copying and pasting the `maxOf3` function definition and renaming it.

Flowchart 1



Flowchart 2



Task 21

Type up and run the *isEven* and *isOdd* functions (code given below).

```
# A function to determine evenness
def isEven(number):
    if (number % 2 == 0):
        return True
    else:
        return False
```

```
# display even numbers < 100
for i in range (100):
    if isEven(i):
        print(i)
```

```
# A function to determine oddness
def isOdd(number):
    return not isEven(number)
```

Task 22

Implement the following two Boolean functions

```
def isEqual(n1, n2):
```

A function to take two values and return `True` if they are both equal; `False` otherwise.

```
def isDifferent(a, b):
```

A function to do the opposite to `isEqual`

Task 23

Type up and run the code below to check if a number is prime or not. Extend the code by adding *function calls* sending both prime and composite numbers to the function.

```
import math

# A function to test whether a number is prime or not
# Returns True if the number is prime; False otherwise
def isPrime(numToCheck):

    # Any number less than 2 is not prime
    if numToCheck < 2:
        return False

    # see if num is evenly divisible by any number up to num/2
    divisor = 2
    while (divisor < numToCheck/2):
        if (numToCheck % divisor == 0):
            return False
        divisor = divisor+1

    # The number must be prime so return True
    return True
```

Task 24

```
# Test harness
for num in range(100):
    if (isPrime(num)):
        print(num, "is prime")
```

A test harness for `isPrime`

a) Predict what the test harness would do?

b) Run the test harness? Was your prediction correct?

c) Investigate. The program looks for factors up to half the number being checked. What would happen if the program stopped at the square root?

d) Modify the test harness so that it displays the first 100 prime numbers.

e) Make a program that displays the n^{th} prime where n is a number entered by the end user.

Task 25

Key in the code below and use it to determine whether the years listed are leap or not.

2000

1900

2017

2012

2100

2400

2600

```
# Determines whether n is leap or not
def isLeap(n):
    if n % 400 == 0:
        return True
    if n % 100 == 0:
        return False
    if n % 4 == 0:
        return True
    else:
        return False

yr = int(input("Enter a year: "))
if isLeap(yr):
   print(yr, "is a leap year")
else:
   print(yr, "is NOT a leap year")
```

Use the code to derive and record some facts about leap years

Describe when is a year a leap year

Task 26

Now examine the two implementations below – one is correct and one contains a subtle error (i.e. one version reports incorrectly that 2000 is not a leap year.)

```
def isLeapV1(year):
    if year % 4 == 0 and year % 100 != 0:
        return True
    elif year % 100 == 0:
        return False
    elif year % 400 ==0:
        return True
    else:
        return False
```

```
def isLeapV2(year):
    if year % 4 == 0 and year % 100 != 0:
        return True
    elif year % 400 ==0:
        return True
    elif year % 100 == 0:
        return False
    else:
        return False
```

Can you explain the subtle error and suggest/implement a solution.

In `isLeapV2` above can you justify the need for the second `elif` block? Explain.

Task 27

Encapsulate the code shown below into two functions – call them `isLeapV3` and `isLeapV4`

```
year = int(input("Enter a year: "))
if year % 4 == 0 and (year % 100 != 0 or year % 400 == 0):
   print(year, "is a leap year")
else:
   print(year, "is NOT a leap year")
```

```
year = int(input("Enter a year: "))

if (year % 4) == 0:
   if (year % 100) == 0:
       if (year % 400) == 0:
           print(year, "is a leap year")
       else:
           print(year, "is NOT a leap year")
   else:
       print(year, "is a leap year")
else:
   print(year, "is NOT a leap year")
```

Compare and contrast the two functions `isLeapV3` and `isLeapV4`.

Task 28

Write a one line Boolean function to determine whether a given year is leap or not