Programming1(Python): Introduction

Assignments Round 2(Repetition Structure)

2.1 Create a program that asks the user if they're bored, until they are. Some examples of how the program works

Bored? (y/n) *n*

Bored? (y/n) *n*

Bored? (y/n) *n*

Bored? (y/n) *n*

Bored? (y/n) *y*

Well, let's stop this, then.

Bored? (y/n) *y*

Well, let's stop this, then.

In this initial version, you may assume the user always answers either "y" or "n". There's no need to worry about incorrect entries. In practice, this means that you're only checking if the user's answer to question is "n", as this is the answer that continues the repetition.

Programming tips:

* Remember that in Python program code, a string including a letter X is marked "X". Instead, mere X without quotation marks refers to a variable named X. If you thus wish to create a variable called answer with the string "X" as a value, enter  
  answer = "X" in Python.
* Your program needs a variable for saving the answer entered by the user. Which value should you use for formatting the variable? (In other words, which original value should you set to the variable?)
* Create a while structure in the program and use its condition to review this variable's value, ie. make the repetition (query) continue if the user answers that they are not bored. Create the while structure so, that the code in the structure is run if the condition is true.
* After the repetition ends, print a final message. In the program code file, this print command should be located after the repetition structure. Use indentation to show the repeated statements (indented inside the repetition structure) and the place where the program implementation continues when the repetition ends (indented on the same level as the word while).
* The program can be implemented so that the same input command is in two different sections of the program code, but also so that the input is only performed once. Can you implement the program so that only one input command is needed?

Which program rows of your program code are executed if you enter the example inputs given in this task as inputs? List the numbers of the performed rows in order. (You don't need to return this task, but should still understand its point.)

a=str(input("Bored? (y/n) "))  
while a=="n":  
 print("Bored? (y/n)",a)  
 a = str(input("Bored? (y/n) "))  
 print("Well, let's stop this, then.")

### 2.2 Bored? (checking for errors)

The program you implemented in the previous section will not work properly when the user enters something other than "y" or "n". Now, implement *a whole new program*, which only contains a repeating structure that asks the user's answer again if the answer is not "y","Y", "n" or "N".

Examples of how the program operates:

Answer Y or N: *q*

Incorrect entry.

Please retry: *w*

Incorrect entry.

Please retry: *n*

You answered n

Answer Y or N: *Y*

You answered Y

Answer Y or N: *N*

You answered N

Programming tips:

* The solution is similar to the previous task's solution, but the condition of the repeating structure is more complicated. Use logical operators to combine the different parts of the condition.

a=str(input("Bored? (y/n) "))  
  
while True:  
 if a=='n':  
  
 a = str(input("Bored? (y/n) "))  
  
 elif a== 'y':  
 print('Well, let\'s stop this, then.')  
 break  
 else:  
 print('Incorrect entry.')  
 a = str(input("Please retry: "))

### 2.3 Bored? (improved version)

Combine the previously implemented programs to create a program that asks the user if they're bored until they are, and additionally requires them to answer with the letter "y", "Y", "n" or "N", ie. asks the user for the answer repeatedly until receiving a valid input.

Examples:

Bored? (y/n) o

Incorrect entry.

Please retry: z

Incorrect entry.

Please retry: m

Incorrect entry.

Please retry: n

Bored? (y/n) n

Bored? (y/n) n

Bored? (y/n) f

Incorrect entry.

Please retry: j

Incorrect entry.

Please retry: y

Well, let's stop this, then.

Bored? (y/n) n

Bored? (y/n) n

Bored? (y/n) y

Well, let's stop this, then.

* Use the [the first task's](https://plus.tuni.fi/tie-02107/spring-2020/kierros02/reading_repetition/tylsaa_a) solution as a basis. Add the verification of the validity of the input, ie. what you implemented [in the second task](https://plus.tuni.fi/tie-02107/spring-2020/kierros02/reading_repetition/tylsaa_b).
* The final result should contain two nested loops (the other loop is "inside" the first loop).

a=str(input("Answer Y or N: "))  
  
if a=='Y' or a=='N':  
 print('You answered ', a)  
  
elif a== 'y' or a=='n':  
 print('You answered ', a)  
  
else:  
 while a!='Y' or a!='N' or a!= 'y' or a!='n':  
 print('Incorrect entry.')  
 a = str(input("Please retry: "))  
 if a=='Y' or a=='N' or a== 'y' or a=='n':  
  
 print('You answered ',a)  
 break

### 2.4. Multiplication table, school version

Create a program that prints a multiplication table for an entered number just like in school, using steps of one to ten.

Example

Choose a number: *6*

1 \* 6 = 6

2 \* 6 = 12

3 \* 6 = 18

...

10 \* 6 = 60

Programming tips:

* When implementing this program, you will have a loop that contains a loop variable. Think about which part of the printing calculation operation uses the loop variable.

### 2.5 Multiplication table, values over a hundred

Create a program that prints a multiplication table for a given number until it reaches a result that is more than hundred.

Example:

Choose a number: *6*

1 \* 6 = 6

2 \* 6 = 12

3 \* 6 = 18

...

17 \* 6 = 102

Programming tips:

* One possible way is calculating the number of needed multiplications at the start of the program. Apart from the calculation, this does not differ too much from the implementation of the program in the previous part and thus teaches nothing new.
* Another possibility, more interesting for learning purposes, is to re-think the type of the loop structure (while or for) and the condition of the loop structure. What needs to be checked on every repetition?

a=int(input("Choose a number: "))  
for x in range(1,100000):  
 b=a\*x;  
 print(x, '\*', a,'=',b)  
 if b>100:  
 break

### 2.5 Number series game Zip Boing

In the game Zip Boing players sit in a ring and call numbers in turns. The first player says 1, the next one 2 and so forth. The game is called Zip Boing because every time the next number is divisible by 3 the player has to say "zip" and every time the number is divisible by 7 the player has to say "boing". Also, if the number is divisible by both the numbers, the player should say "zip boing".

Implement a program that makes a cheat sheet for as many numbers as the player wants. The program should first ask how many numbers the cheat sheet should include and then print them in correct order (with all the zips, boings and zip boings).

Example:

How many numbers would you like to have? 10

1

2

zip

4

5

zip

boing

8

zip

10

Advice:

* You need a repetition structure that goes from 1 to the given maximum value. You also need a control structure that prints out either zip, boing, zip boing or the number itself.
* Think about the order of the structures: which is the correct order for the control structure, i.e., what should happen first etc.
* The divisibility of a number may be tested with the divisibility operator % which has been tried out during the previous round (giving change) and in the lecture.

a=int(input("How many numbers would you like to have? "))  
for x in range(1,a+1):  
  
 if x%3==0 and x%7==0:  
 print("zipboing")  
  
 else:  
 if x % 3 == 0:  
 print("zip")  
 elif x % 7 == 0:  
 print("boing")  
 else:  
 print(x)

### 2.6 Fixing the spaces

Create a program that asks the user's name and then greets them with the text in the example run below. We intend you to format the printout exactly as in the example printout, ie. so that that the comma printed after the user's name is attached to the name without the name and the comma being separated with a space.

An example of how the program operates:

Tell us your name: *Teemu*

Hey Teemu, the printout formatting is going well!

Programming tips:

* Check the previous materials to see how to define the print command separator character.
* Remember that an empty character, ie. " ", is different from the empty string "".

a=str(input("Tell us your name: "))  
print('Hey ',a,',','the printout formatting is going well!')

### 2.7 Fixing specificities

[The attached file](https://plus.tuni.fi/graderB/static/tie02101-s2020/static/code/tulostuksenmuotoilu_b.py) contains a program that prints the approximate value of the pi. Format the program's printout so that in the first section, the approximate value of the pi is printed to the specificity of zero decimals (ie. as an integer) and, in the second section, to the specificity of four decimals. The program's printout should thus look like this:

The approximate value of pi is 3 or, if we want to get specific, 3.1416

Programming tips:

* The formatting of the printing specificity of the printed number happens using format function of string.
* You should not use any other functions than format to help in rounding out numbers. When printing happens using the formatting function, the rounding is also automatically performed correctly. You should also perform the rounding needed in the forthcoming course tasks using the format function. You should specifically NOT use the Python's round function for the course tasks, as it changes the value of the number and thus means that the calculations will be inexact. **This causes issues for the automatic verification process.**
* Remember to think about the difference between numeric values and printing specificity!

### 2.8 Fixing field width

[The attached file](https://plus.tuni.fi/graderB/static/tie02101-s2020/static/code/tulostuksenmuotoilu_c.py) includes a program that prints a simple cheat sheet for learning the multiplication table. Format the program's printout so that each multiplication result is shown in a printout field, which is four characters wide. The program's printout should thus look like this:

1 2 3 4 5 6 7 8 9 10

2 4 6 8 10 12 14 16 18 20

3 6 9 12 15 18 21 24 27 30

4 8 12 16 20 24 28 32 36 40

5 10 15 20 25 30 35 40 45 50

6 12 18 24 30 36 42 48 54 60

7 14 21 28 35 42 49 56 63 70

8 16 24 32 40 48 56 64 72 80

9 18 27 36 45 54 63 72 81 90

10 20 30 40 50 60 70 80 90 100

Programming tips:

* Note that since this task includes practicing the printing of spaces, the automatic evaluation also checks the amount of the spaces, which must to be exactly the correct one. (In other tasks, the automatic verification does not take the amount of spaces into account, so it doesn't mind if there is one space between the words instead of two spaces, and so on.)

def main():  
 for i in range(1,11):  
 for j in range(1,11):  
 if i\*j<10:  
 print(i\*j, " ", end="")  
 else:  
 print(i \* j, " ", end="")  
 print()  
  
main()

### 2.9 Fibonacci series

The two first numbers of the Fibonacci sequence are ones. The numbers after them are calculated by counting together the two preceding numbers, again and again. Implement a program that prints the Fibonacci sequence for a number of times set by the user:

How many Fibonacci numbers do you want? *7*

1. 1

2. 1

3. 2

4. 3

5. 5

6. 8

7. 13

Programming tips:

* As in the previous task, the user will know the number of the repeats that must be made at the start of the repeating structure. You thus already know how to implement the repeating structure.
* A visual presentation of counting Fibonacci numbers can be seen here [from a video](https://www.youtube.com/watch?v=AzfHbETDbN8#t=057) (about 20 seconds of it are related to the task. The entire video is worth watching for educational purposes, of course).
* The calculation of Fibonacci numbers can thus use three variables, one for saving the previous calculated number, one for saving the one preceeding it, and one using the new number calculated in the repetition structure’s loop using the two previous variables.
* When calculating the new number, the variable values are received from the variables calculated in the previous round by transferring them in the way represented by the next image:  
  A close up of a logo

  Description automatically generated  
  Also consider which transfer to perform first and which one next.
* The calculation of the start of the string must be considered carefully. Can you set the variables so that the calculation works using the same principles every round? Or must the initial rounds be processed differently from the others? (The previous section of the material discussed a selection command inside a repeating structure.)

i=int(input("How many Fibonacci numbers do you want? "))  
a=0;  
b=1;  
if i<0:  
 print("incorrect input")  
else:  
 for x in range(1,i+1):  
 if x == 1:  
 print(x, '.', b)  
 else:  
 c=a+b  
 a=b  
 b=c  
 print(x, '.', c)

### 2.10 The dates of the year

Create a program that prints all of this year’s dates as follows:

1.1.

2.1.

3.1.

4.1.

5.1.

...

31.1.

1.2.

2.2.

3.2.

...

28.2.

1.3.

...

Programming tips:

* The execution of the program proceeds roughly like this: Start with month 1 and day 1. Increase the day by steps of one until it is 31, printing all dates (repetition). After this, increase the month by one and take 1 as the date again. Then increase the date and print again (same repetition as before). Thinking this through, you will note that repetition also happens when the month is increased: after all dates for each month are printed, the month increases by one. A condition also exists for ending the repetition. Which condition?
* Thus, there are two repeating structures! How are they related to each other? Also remember [the task that added a check for errors to a query repetition structure](https://plus.tuni.fi/tie-02107/spring-2020/kierros02/reading_repetition/tylsaa_c/).
* As months have a different amount of dates, you need decision structures, in addition to repeating structures (for the first round).
* In the printout, the numbers of day and month must not include spaces between the numbers and dots in order for the automatic verification to approve the program. (In brief: Add sep="" to the print command.)

a=2020;  
  
for i in range(1,13):  
 if i==1 or i==3 or i==7 or i==8 or i==10 or i==12:  
 for j in range(1,32):  
 print(j,'.',i,'.',a)  
 elif i==4 or i==5 or i==6 or i==9 or i==11:  
 for j in range(1,31):  
 print(j,'.',i,'.',a)  
 else:  
 for j in range(1, 29):  
 print(j, '.', i, '.', a)

### 2.11 TGIF

Create a program that prints the dates for all the Fridays in 2014. In 2014, the first Friday was 3.1.

Programming tips:

* One possible solution is based on every seventh day being Friday. In the previous task, you implemented a program that goes through all the dates of a year. You can add this program a counter whose value is always increased by one when the program goes through the dates. The calculator can then be used to create a decision structure, which only prints on every seventh date.
* Another solution is increasing the date in increments of seven and using a remainder operator to select the right date when a month changes.

a=2014;  
c = 3;  
b=7;  
for i in range(1,13):  
  
 if i==1 or i==3 or i==5 or i==7 or i==8 or i==10 or i==12:  
 for j in range(c,32,7):  
 print(j, '.', i, '.', sep="")  
 c = 7-(31-j);  
 elif i==4 or i==6 or i==9 or i==11:  
 for j in range(c,31,7):  
 print(j, '.', i, '.', sep="")  
 c = 7 - (30 - j);  
 else:  
 for j in range(c, 29,7):  
 print(j, '.', i,'.', sep="")  
 c = 7 - (28 - j);