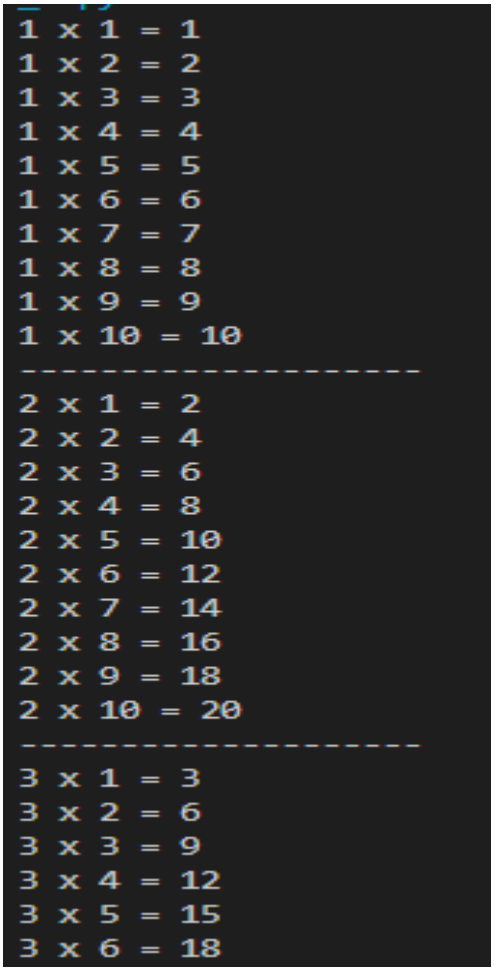
1. في هذا المشروع سوف نقوم بمساعدة أنفسنا على مراجعة جدول الضرب، ولكن أحد الحلول التي قد تخطر على بالك كمبرمج هو عرض جدول ضرب خاص بك كما بالصورة التالية

Make your own Multiplication table like the following.



{//}

إذا لم توفق للوصول للحل يمكنك السعي مرة أخرى بمساعدة الخطوات التالية

```
# 1. multiplication table in the following format:

# 1 x 1 = 1

# 1 x 2 = 2

# ........

# loop 1 to 10 for each table like table 1, table 2, table 3, etc.

# loop 1 to 10 for each table like 1 x 1, 1 x 2, 1 x 3, etc.

# calculate the product

# print the product in the format of 1 x 1 = 1

# print a line to separate the tables
```

{//}

```python
# 1. multiplication table in the following format:
# 1 x 1 = 1
# 1 x 2 = 2
# ........

# loop 1 to 10 for each table like table 1, table 2, table 3, etc
for i in range(1, 11):
    # loop 1 to 10 for each table like 1 x 1, 1 x 2, 1 x 3, etc
    for j in range(1, 11):
        # calculate the product
        product = i * j
        # print the product in the format of 1 x 1 = 1
        print(f"{i} x {j} = {product}")

    # print a line to separate the tables
    print("-"*20)
```

{//}

2.قم بجعل كل كلمات القائمة التالية uppercase

Make All the following list items uppercase.

```
words = ['have', 'that', 'they', 'with', 'this', 'from',
'which', 'would', 'will', 'there', 'make', 'when', 'more',
'other', 'what', 'time', 'about', 'than', 'into', 'could',
'state', 'only', 'year', 'some', 'take', 'come', 'these',
'know', 'like', 'then', 'first', 'work', 'such', 'give',
'over', 'think', 'most', 'even', 'find', 'also', 'after',
'many', 'must', 'look',
'before', 'great', 'back', 'through', 'long', 'where', 'much',
'should', 'well', 'people', 'gouda', 'just', 'because', 'good',
'each', 'those', 'feel', 'seem', 'high', 'place', 'little',
'world', 'very', 'still', 'nation', 'hand', 'life', 'tell',
'write', 'become', 'here', 'show', 'house', 'both', 'between',
'need', 'mean', 'call', 'develop', 'under', 'last', 'right',
'move', 'thing', 'general', 'school', 'never', 'same',
'another', 'begin', 'while', 'number', 'part', 'turn', 'real',
'leave', 'might', 'want', 'point', 'form', 'child', 'small',
'since', 'against', 'late', 'home', 'interest', 'large',
'person', 'open', 'public', 'follow', 'during', 'present',
'without', 'again', 'hold', 'codezilla', 'govern', 'around',
'head', 'consider', 'word', 'program', 'problem', 'however',
'lead', 'system', 'order', 'plan', 'keep', 'face', 'group',
'play', 'stand', 'increase', 'early', 'course', 'change',
'help', 'line', 'possible', 'fact', 'down']
```

إذا لم توفق للوصول للحل يمكنك السعي مرة أخرى بمساعدة الخطوات التالية

```
# 2. modify this list of words to make All words are uppercase

# list of words

# loop through the list of words

# convert the current word to uppercase

# print the modified list of words
```

{//}

```
# 2. modify this list of words to make All words are uppercase

# list of words
words = [
        'have', 'that', 'they', 'with', 'this', 'from',
'which', 'would', 'will', 'there',
        'make', 'when', 'more', 'other', 'what', 'time',
'about', 'than', 'into', 'could',
        'state', 'only', 'year', 'some', 'take', 'come',
'these', 'know', 'like', 'then',
        'first', 'work', 'such', 'give', 'over', 'think',
'most', 'even', 'find', 'also',
        'after', 'many', 'must', 'look', 'before', 'great',
'back', 'through', 'long',
        'where', 'much', 'should', 'well', 'people', 'gouda',
'just', 'because', 'good',
        'each', 'those', 'feel', 'seem', 'high', 'place',
'little', 'world', 'very', 'still',
        'nation', 'hand', 'life', 'tell', 'write', 'become',
'here', 'show', 'house', 'both',
        'between', 'need', 'mean', 'call', 'develop', 'under',
'last', 'right', 'move', 'thing',
        'general', 'school', 'never', 'same', 'another',
'begin', 'while', 'number', 'part',
        'turn', 'real', 'leave', 'might', 'want', 'point',
'form', 'child', 'small', 'since',
        'against', 'late', 'home', 'interest', 'large',
'person', 'open', 'public', 'follow',
        'during', 'present', 'without', 'again', 'hold',
'codezilla', 'govern', 'around',
        'head', 'consider', 'word', 'program', 'problem',
'however', 'lead', 'system',
        'order', 'plan', 'keep', 'face', 'group', 'play',
'stand', 'increase',
```

```
        'early', 'course', 'change', 'help', 'line',
'possible', 'fact', 'down'
        ]


# loop through the list of words
for i in range(len(words)):
    # convert the current word to uppercase
    words[i] = words[i].upper()

# print the modified list of words
print(words)
```

3. في القائمة التالية قم بتحويل كل inner list إلى جملة ثم قم بتخزين هذه القائمة داخل متغير sentences وقم بطباعته، وأيضا قم بعمل قائمة جديدة تحتوي على نفس عناصر قائمة sentences ولكن مع استبدال المسافات ب الشرط (-) أو dashes، وجعل كل الحروف uppercase

Do the following:

A. convert each inner list to a string and join them with a space and add them to a list named sentences.

B. Make another list replace spaces with dashes and convert each word to uppercase.

```
words = [["Hello", "from", "Codezilla"],
    ["Python", "Course", "is", "awesome"],
    ["I", "enjoy", "learning", "Python", "with", "Codezilla"]]
```

{//}

إذا لم توفق للوصول للحل يمكنك السعي مرة أخرى بمساعدة الخطوات التالية

```
# 3. Do the following:

# A. convert each inner list to a string and join them with a
space and add them to a list named sentences

# B. Make another list named modified_sentences replace spaces
with dashes and convert each word to uppercase

# make a list of sentences

# convert each inner list to a string and join them with a
space

# make another list named modified_sentences

# Loop through the list of sentences

# Replace Spaces with dashes and convert to uppercase

# Print the sentences and modified_sentences
```

{//}

```
# 3. Do the following:
# A. convert each inner list to a string and join them with a
space and add them to a list named sentences
# B. Make another list named modified_sentences replace spaces
with dashes and convert each word to uppercase

words = [["Hello", "from", "Codezilla"],
    ["Python", "Course", "is", "awesome"],
    ["I", "enjoy", "learning", "Python", "with", "Codezilla"]]

# make a list of sentences
sentences = []
for lst in words:
    # convert each inner list to a string and join them with a
space
    sentences.append(" ".join(lst))


# make another list named modified_sentences
modified_sentences = []

# Loop through the list of sentences
for i in range(len(sentences)):
    # Replace Spaces with dashes and convert to uppercase
    modified_sentences.append(sentences[i].replace(" ", "-
").upper())



# Print the sentences and modified_sentences
print(f"The list of sentences is: {sentences}")
print(f"The modified list of sentences is:
{modified_sentences}")
```

4.قم بإيجاد أصغر رقم في القائمة التالية دون استخدام min & sort

Find the smallest number in the following list without using min and sort.

```
numbers = [-588, -479, -713, -701, -885, -578, -835, -466, -935, -665, -360, -837, -389, -367, -454, -668, -907, -822, -541, -680, -405, -330, -625, -916, -331, -876, -689, -753, -810, -648, -787, -952, -718, -401, -502, -699, -533, -450, -580, -725]
```

إذا لم توفق للوصول للحل يمكنك السعي مرة أخرى بمساعدة الخطوات التالية

```
# 4. Min number in a list without using min() and sort()

# assume the first number is the smallest

# loop through the list of numbers

# check if the current number is smaller than the smallest

# if so, make the current number the smallest
```

{//}

```python
# 4. Min number in a list without using min() and sort()


numbers = [-588, -479, -713, -701, -885, -578, -835, -466, -
935, -665, -360, -837, -389, -367, -454, -668, -907, -822, -
541, -
           680, -405, -330, -625, -916, -331, -876, -689, -753,
-810, -648, -787, -952, -718, -401, -502, -699, -533, -450, -
580, -725]

# assume the first number is the smallest
smallest = numbers[0]

# loop through the list of numbers
for number in numbers:
    # check if the current number is smaller than the smallest
    if number < smallest:
        # if so, make the current number the smallest
        smallest = number

print(f"The smallest number is {smallest}")
```

5.قم بإيجاد أصغر رقم فردي في القائمة التالية دون استخدام min & sort

Find the smallest, odd number in the following list without using min and sort.

```
numbers = [-500, -694, -762, -445, -348, -361, -758, -594, -954, -861, -610, -549, -336, -400, -600, -836, -671, -573, -555, -390, -450, -811, -849, -870, -815, -694, -951, -588, -484, -609, -674, -411, -408, -498, -649, -541, -441, -839, -567, -898]
```

{//}

إذا لم توفق للوصول للحل يمكنك السعي مرة أخرى بمساعدة الخطوات التالية

```
# 5. Min odd number in a list without using min() and sort()

# assume the first odd number is the smallest

# check if the current number is odd

# if so, make the current number the smallest

# break the loop to avoid checking the rest of the numbers,
since we already get the first odd number

# loop through the list of numbers

# check if the current number is smaller than the smallest and
odd

# if so, make the current number the smallest
```

{//}

```
# 5. Min odd number in a list without using min() and sort()


numbers = [-500, -694, -762, -445, -348, -361, -758, -594, -
954, -861, -610, -549, -336, -400, -600, -836, -671, -573, -
555, -
           390, -450, -811, -849, -870, -815, -694, -951, -588,
-484, -609, -674, -411, -408, -498, -649, -541, -441, -839, -
567, -898]

# assume the first odd number is the smallest
for number in numbers:
    # check if the current number is odd
    if number % 2 != 0:
        # if so, make the current number the smallest
        smallest_odd = number
        # break the loop to avoid checking the rest of the
numbers, since we already get the first odd number
        break

# loop through the list of numbers
for number in numbers:
    # check if the current number is smaller than the smallest
and odd
    if number % 2 != 0 and number < smallest_odd:
        # if so, make the current number the smallest
        smallest_odd = number


print(f"The smallest odd number is {smallest_odd}")
```

6.قم بإيجاد متوسط طول الكلمة في النص التالي

Find the average word length in the following sentence.

```
sentence = """Python is a widely used high-level
general-purpose interpreted dynamic programming language
Its design philosophy emphasizes code readability and its
syntax allows programmers to express concepts in fewer lines of
code
than possible in many other languages
The language provides constructs intended to enable clear
programs on both a small and large scale
"""
```

إذا لم توفق للوصول للحل يمكنك السعي مرة أخرى بمساعدة الخطوات التالية

```
# 6. Average word length

# split the sentence into a list of words

# initialize the total length

# loop through the list of words

# increment the total length

# calculate the average word length

# print the average word length
```

{//}

```python
# 6. Average word length


sentence = """Python is a widely used high-level general-
purpose interpreted dynamic programming language
Its design philosophy emphasizes code readability and its
syntax allows programmers to express concepts in fewer lines of
code
than possible in many other languages
The language provides constructs intended to enable clear
programs on both a small and large scale
"""

# split the sentence into a list of words
words_list = sentence.split()

# initialize the total length
total_length = 0

# loop through the list of words
for word in words_list:
    # increment the total length
    total_length += len(word)

# calculate the average word length
average_length = total_length / len(words_list)

# print the average word length
print(f"The average word length is {average_length:.2f}
characters.")
```

{//}

7.قم بإصلاح الخطأ في الكود التالي بطريقتين على الأقل حتى يكون

Found Codezilla!

أخر شيء يتم طباعته (من المهم أن تلقي نظرة على الإجابات)

Debug the following code with at least 2 solutions, so that "Found Codezilla!" is the last printed output.

```
nested_list = [["Hello", "from", "Codezilla"],
    ["Python", "Course", "is", "awesome"],
    ["I", "enjoy", "learning", "Python"]]


for lst in nested_list:
    print(lst) # You can not remove this line
    for word in lst:
        print(word) # You can not remove this line
        if word == "Codezilla":
            print("Found Codezilla!") # This should be the last
line to be printed
            break
```

{//}

إذا لم توفق للوصول للحل يمكنك السعي مرة أخرى بمساعدة الخطوات التالية

```
# 7. debug the following code with at least 2 solutions, so
that print("Found Codezilla!") is the last printed output

##########

"""

The problem was like we have seen is that the program was
exiting the inner loop and not the outer loop,

so the program continued after break.

"""

"""

Important: the exit() function is used to exit the program, so
nothing after exit() will be executed.

if you tried to run the 3 solutions at the same time, you will
see that the program will exit after the first one.

This is make the last solution the best one.

As if we want to do something after the loop, we can not use
exit() function.
"""
```

```
# Solution 1: comparing words and exit the program with exit()
function



##########

# Solution 2: comparing words and exit the program with
continue and exit() function

##########

# Solution 3: checking for words with in and exit the loop with
break (the best solution)
```

{//}

```
# 7. debug the following code with at least 2 solutions, so
that "Found Codezilla!" is the last printed output
nested_list = [["Hello", "from", "Codezilla"],
    ["Python", "Course", "is", "awesome"],
    ["I", "enjoy", "learning", "Python"]]


for lst in nested_list:
    print(lst) # You can not remove this line
    for word in lst:
        print(word) # You can not remove this line
        if word == "Codezilla":
            print("Found Codezilla!") # This should be the last
line to be printed
            break
##########

"""
The problem was like we have seen is that the program was
exiting the inner loop and not the outer loop,
so the program continued after break.
"""

"""
Important: the exit() function is used to exit the program, so
nothing after exit() will be executed.
if you tried to run the 3 solutions at the same time, you will
see that the program will exit after the first one.
This is make the last solution the best one.
As if we want to do something after the loop, we can not use
exit() function.
"""
```

{//}

```
# Solution 1: comparing words and exit the program with exit()
function


for lst in nested_list:
    print(lst)
    for word in lst:
        print(word)
        if word == "Codezilla":
            print("Found Codezilla!")

         exit() # EXIT THE WHOLE PROGRAM




##########
# Solution 2: comparing words and exit the program with
continue and exit() function

for lst in nested_list:
    print(lst)
    for word in lst:
        print(word)
        if word != "Codezilla":
            continue
        else:
            print("Found Codezilla!")
            exit()  # EXIT THE WHOLE PROGRAM
```

{//}

```
##########

# Solution 3: checking for words with in and exit the loop with
break (the best solution)

for lst in nested_list:
    print(lst)
    if "Codezilla" in lst:
        print("Found Codezilla!")
        break  # EXIT THE LOOP ONLY
```