Operations with `for` Loop

Operations with the `for` Loop(review)

- In the following example, you'll get a **number** from the user and *print* **a sentence** the *number of times* we take from the user :

```
1  times = int(input("How many times should I say 'I love you'"))
2
3  for i in range(times):
4      print('I love you')
5
```

Operations with the `for` Loop(review)

- In the following example, you'll get a number from the user and print a sentence the number of times we receive from the user :

```
1  times = int(input("How many times should I say 'I love you'"))
2
3  for i in range(times):
4      print('I love you')
5
```

Let's say the user enters **3**.

```
1  I love you
2  I love you
3  I love you
4
```

Operations with the `for` Loop

- **Task :** This time, write a code block that asks the user a number between 1 and 10 then puts that number into the multiplication table.
- For example, the output for 5 should be as follows :

```
5x0 =  0
5x1 =  5
5x2 =  10
5x3 =  15
5x4 =  20
5x5 =  25
5x6 =  30
5x7 =  35
5x8 =  40
5x9 =  45
5x10 =  50
```

Working with the Iterators

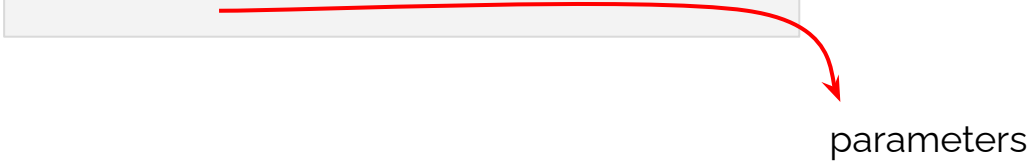
- **The output can be like :**

```
1  nmbr = int(input('enter a number between 1-10'))
2
3  for i in range(11):
4      print('{}x{} = '.format(nmbr, i), nmbr * i)
5
```

Operations with the `for` Loop(review)

- Let's take a close look at the `range()` function.
  - As we stated before, the formula syntax of the `range()` function is :

```
range(start, stop, step)
```



Operations with the `for` Loop(review)

- (... continued)
  - Consider this example :

```
1  b = list(range(11))
2
3  print(b)
4
```

Operations with the `for` Loop(review)

- Let's take a close look at the `range()` function.
  - It creates an iterable sequence of numbers. And it can be simply converted into the `list`, `set`, and `tuple`.
  - Consider this example :

```
1  b = list(range(11))
2
3  print(b)
4
```

```
1  [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
2
```

Operations with the `for` Loop(review)

- (... continued)
  - Here's the other examples :

```
1  a = set(range(0,10))
2
3  print(a)
4
```

Operations with the `for` Loop(review)

- (... continued)
  - Here's the other examples :

```
1  a = set(range(0,10))
2
3  print(a)
4
```

```
1  {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
2
```

Operations with the `for` Loop(review)

- (... continued)
  - Here's the other examples :

```
1  a = set(range(0,10))
2
3  print(a)
4
```

```
1  {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
2
```

```
1  c = tuple(range(11))
2
3  print(c)
4
```

Operations with the `for` Loop(review)

- (... continued)
  - Here's the other examples :

```
1  a = set(range(0,10))
2
3  print(a)
4
```

```
1  {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
2
```

```
1  c = tuple(range(11))
2
3  print(c)
4
```

```
1  (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
2
```

Operations with the `for` Loop (review)

- An asterisk 👉 `*` separates the elements of the iterables.
  - Let's take a look at an example of the `range()` function with starred 👉 `*` expression :

```
1  print(range(5))  # it will not print the numbers in sequence
2
3  print(*range(5))  # '*' separates its elements
4
```

What is the output? Try to figure out in your mind...

Operations with the `for` Loop(review)

- (... continued)
  - Let's take a look at an example of the `range()` function with starred 👉 `*` expression :

```
1  print(range(5))  # it will not print the numbers in sequence
2
3  print(*range(5))  # '*' separates its elements
4
```

```
1  range(0, 5)
2  0 1 2 3 4
3
```

Operations with the `for` Loop(review)

- (... continued)
  - Here's another example of the `range()` function with starred 👉 `*` expression :

```
1  print(*range(5,25,2))
2
```

What is the output? Try to figure out in your mind...

Operations with the `for` Loop(review)

- (... continued)
  - Here's another example of the `range()` function with starred 👉 `*` expression :

```
1  print(*range(5,25,2))
2
```

```
1  5 7 9 11 13 15 17 19 21 23
2
```

Operations with the `for` Loop(review)

- (... continued)
  - Starred 👉 `*` expression can also be used to separate the other **iterable** objects. Such as `str` :

```
1  print(*('separate'))
2
```

Operations with the `for` Loop(review)

- (... continued)
  - Starred 👉 `*` expression can also be used to separate the other **iterable** objects. Such as `str` :

```
1  print(*('separate'))
2
```

```
1  s e p a r a t e
2
```

Operations with the `for` Loop(review)

- (... continued)
  - You can create reverse sequence numbers using a negative `step`.

```
1  print(*range(10,0,-2))
2
```

What is the output? Try to figure out in your mind...

Operations with the `for` Loop(review)

- (... continued)
  - You can create reverse sequence numbers using a negative `step`.

```
1  print(*range(10,0,-2))
2
```

```
1  10 8 6 4 2
2
```

Operations with the `for` Loop(review)

- Multiple variables in `for` loop.
  - Examine this example carefully :

```
zip(iterator1, iterator2, ...)
```

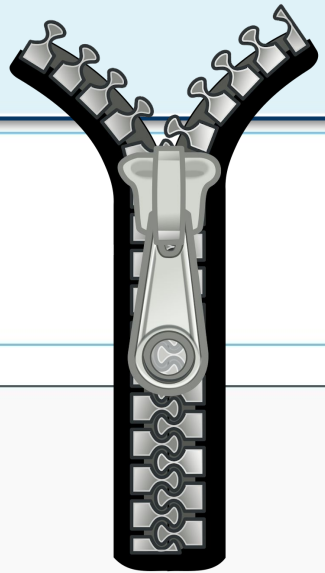
```
1  text = ['one','two','three','four','five']
2  numbers = [1, 2, 3, 4, 5]
3  for x, y in zip(text, numbers):
4      print(x, ':', y)
5
```

Use your IDEs

Operations with the `for` Loop (review)

Tips :

- `zip()` function make an iterator that aggregates elements from each of the iterables.



```
text = ['one','two','three','four','five']
numbers = [1, 2, 3, 4, 5]
for x, y in zip(text, numbers):
    print(x, ':', y)

```

```
one : 1
two : 2
three : 3
four : 4
five : 5

```

Operations with the `for` Loop

- **Task** : **Python Program to collect the odd and even numbers in two different `lists`.**
  - Write a program to choose and collect the even and odd numbers (1 to 10) in two different `list`.
  - Print the result such as :

```
evens: [0, 2, 4, 6, 8]
odds : [1, 3, 5, 7, 9]
```

Operations with the `for` Loop

- **The code might be like** :

```
1  evens = []
2  odds = []
3
4  for n in range(10):
5      if n % 2 == 0:
6          evens.append(n)
7      else:
8          odds.append(n)
9
10 print(evens)
11 print(odds)
```

Output

```
[0, 2, 4, 6, 8]
[1, 3, 5, 7, 9]
```

Operations with the `for` Loop

- **Task** : **Python Program to sum the amount of odd and even numbers in a `tuple`/`list`.**
  - Write a code that counts the odd and even numbers in a given **`list`** or **`tuple`**.
  - Print the result such as :

```
example list: [11, 2, 24, 61, 48, 33, 3]
example output : The number of even numbers : 3
                 The number of odd numbers : 4
```

Operations with the `for` Loop

- **The code might be like** :

```
1  numbers = (11, 36, 33, 66, 89, 21, 32, 16, 10)
2  odds = 0
3  evens = 0
4  for i in numbers:
5      if not i % 2:
6          evens+=1
7      else:
8          odds+=1
9  print("The number of even numbers :", evens)
10 print("The number of odd numbers  :", odds)
11
```

Output

```
The number of even numbers : 5
The number of odd numbers  : 4
```

Operations with the `for` Loop

- **Task** : **Python Program to print out the numbers.**
  - Using the `for` loop, print the numbers from **1** to **9** as many as it is and get the following output.

```
1
22
333
4444
55555
666666
7777777
88888888
999999999
```

Operations with the `for` Loop

- **The code might be like** :

```
1 for i in range(1, 10):
2     print(str(i) * i)
3
```

Operations with the `for` Loop

- **Task** : **Python Program to sum of the numbers from 1 to 74**
  - Get the output of **2775** as a sum of the numbers between **1** - **74** (including).
  - Use `for` loop to make this calculation.

Operations with the `for` Loop

- **The code might be like** :

```
1  sum_num=0
2
3  for i in range(1, 75):
4      sum_num += i
5
6  print(sum_num)
7
```

Nested `for` Loop

Nested `for` Loop (review)

- Simple structure of the nested `for` loops look like :

```
for variable1 in iterable1:
    for variable2 in iterable2:
        body
```

Nested for Loop (review)

- Consider this example of the nested for loop :

```
1  who = ['I am ', 'You are ']
2  mood = ['happy', 'confident']
3  for i in who:
4      for ii in mood:
5          print(i + ii)
6
```

Nested for Loop

- Consider this example of the nested for loop :

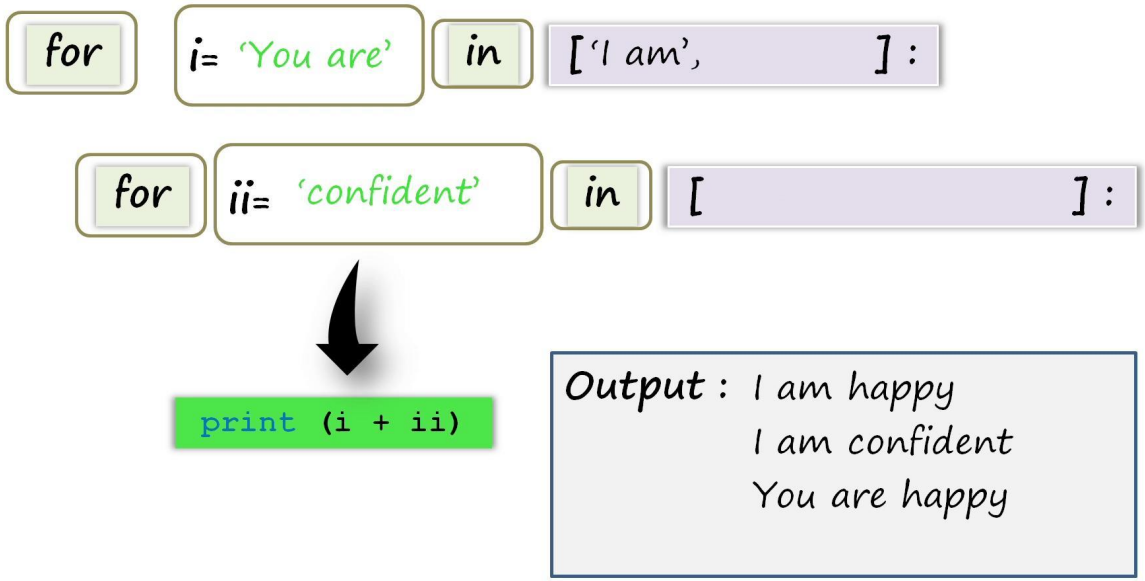
```
1  who = ['I am ', 'You are ']
2  mood = ['happy', 'confident']
3  for i in who:
4      for ii in mood:
5          print(i + ii)
6
```

First *outer* **then** *inner* loop runs.

```
1  I am happy
2  I am confident
3  You are happy
4  You are confident
5
```

Nested for Loop (review)

- You can follow the animated diagram of this nested `for` loop for a better understanding.



Nested `for` Loop

- **Task** : **Concatenation string elements from two separate `lists`.**
  - Write a code that takes string elements one by one and prints a sentence using nested `for` loops :
  - The given lists and sample outputs are :

```
names = ["susan", "tom", "edward"]
mood = ["happy", "sad"]
example output : susan is happy
                 susan is sad
                 tom is happy
                      .
                      .
```

Nested for Loop

- **The code might be like** :

```
1  names = ["susan", "tom", "edward"]
2  mood = ["happy", "sad"]
3
4  for i in names:
5      for ii in mood:
6          print(i + " is " + ii)
7
```

Output

```
susan is happy
susan is sad
tom is happy
tom is sad
edward is happy
edward is sad
```