30 Days Of Python: Day 10 - Loops

Loops

Life is full of routines. In programming we also do lots of repetitive tasks. In order to handle repetitive task programming languages use loops. Python programming language also provides the following types of two loops:

1. while loop
2. for loop

While Loop

We use the reserved word *while* to make a while loop. It is used to execute a block of statements repeatedly until a given condition is satisfied. When the condition becomes false, the lines of code after the loop will be continued to be executed.

```
  # syntax
while condition:
    code goes here
```

Example:

```
count = 0
```

```
while count < 5:
    print(count)
    count = count + 1
#prints from 0 to 4
```

In the above while loop, the condition becomes false when count is 5. That is when the loop stops. If we are interested to run block of code once the condition is no longer true, we can use *else*.

```
  # syntax
while condition:
    code goes here
else:
    code goes here
```

Example:

```
count = 0
while count < 5:
    print(count)
    count = count + 1
else:
    print(count)
```

The above loop condition will be false when count is 5 and the loop stops, and execution starts the else statement. As a result 5 will be printed.

Break and Continue - Part 1

- Break: We use break when we like to get out of or stop the loop.

```
# syntax
while condition:
    code goes here
    if another_condition:
        break
```

Example:

```
count = 0
while count < 5:
    print(count)
    count = count + 1
    if count == 3:
        break
```

The above while loop only prints 0, 1, 2, but when it reaches 3 it stops.

- Continue: With the continue statement we can skip the current iteration, and continue with the next:

```
  # syntax
while condition:
    code goes here
    if another_condition:
```

```
        continue
```

Example:

```
count = 0
while count < 5:
    if count == 3:
        count = count + 1
        continue
    print(count)
    count = count + 1
```

The above while loop only prints 0, 1, 2 and 4 (skips 3).

For Loop

A *for* keyword is used to make a for loop, similar with other programming languages, but with some syntax differences. Loop is used for iterating over a sequence (that is either a list, a tuple, a dictionary, a set, or a string).

- For loop with list

```
# syntax
for iterator in lst:
    code goes here
```

Example:

```
numbers = [0, 1, 2, 3, 4, 5]
for number in numbers: # number is temporary name to refer to the list's items,
valid only inside this loop
    print(number)       # the numbers will be printed line by line, from 0 to 5
```

- For loop with string

```
# syntax
for iterator in string:
    code goes here
```

Example:

```
language = 'Python'
for letter in language:
    print(letter)


for i in range(len(language)):
    print(language[i])
```

- For loop with tuple

```
# syntax
for iterator in tpl:
    code goes here
```

Example:

```
numbers = (0, 1, 2, 3, 4, 5)
for number in numbers:
    print(number)
```

- For loop with dictionary Looping through a dictionary gives you the key of the dictionary.

```
  # syntax
for iterator in dct:
    code goes here
```

Example:

```
person = {
    'first_name':'Suniksha',
    'last_name':'Patel',
    'age':20,
    'country':'Finland',
    'is_marred':False,
    'skills':['JavaScript', 'React', 'Node', 'MongoDB', 'Python'],
    'address':{
        'street':'Space street',
        'zipcode':'02210'
    }
}
for key in person:
    print(key)

for key, value in person.items():
    print(key, value) # this way we get both keys and values printed out
```

- Loops in set

```
# syntax
for iterator in st:
    code goes here
```

Example:

```
it_companies = {'Facebook', 'Google', 'Microsoft', 'Apple', 'IBM', 'Oracle',
'Amazon'}
for company in it_companies:
    print(company)
```

Break and Continue - Part 2

Short reminder: *Break*: We use break when we like to stop our loop before it is completed.

```
# syntax
for iterator in sequence:
    code goes here
    if condition:
```

```
        break
```

Example:

```
numbers = (0,1,2,3,4,5)
for number in numbers:
    print(number)
    if number == 3:
        break
```

In the above example, the loop stops when it reaches 3.

Continue: We use continue when we like to skip some of the steps in the iteration of the loop.

```
  # syntax
for iterator in sequence:
    code goes here
    if condition:
        continue
```

Example:

```
numbers = (0,1,2,3,4,5)
for number in numbers:
    print(number)
    if number == 3:
        continue
    print('Next number should be ', number + 1) if number != 5 else print("loop's
end") # for short hand conditions need both if and else statements
print('outside the loop')
```

In the example above, if the number equals 3, the step *after* the condition (but inside the loop) is skipped and the execution of the loop continues if there are any iterations left.

The Range Function

The *range()* function is used list of numbers. The *range(start, end, step)* takes three parameters: starting, ending and increment. By default it starts from 0 and the increment is 1. The range sequence needs at least 1 argument (end). Creating sequences using range

```
lst = list(range(11))
print(lst) # [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
st = set(range(1, 11))    # 2 arguments indicate start and end of the sequence,
step set to default 1
print(st) # {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}

lst = list(range(0,11,2))
print(lst) # [0, 2, 4, 6, 8, 10]
st = set(range(0,11,2))
print(st) #  {0, 2, 4, 6, 8, 10}

# syntax
```

```
for iterator in range(start, end, step):
```

Example:

```
for number in range(11):
    print(number)   # prints 0 to 10, not including 11
```

Nested For Loop

We can write loops inside a loop.

```
# syntax
for x in y:
    for t in x:
        print(t)
```

Example:

```
person = {
    'first_name': 'Suniksha',
    'last_name': 'Patel',
    'age': 20,
    'country': 'Finland',
    'is_marred': False,
    'skills': ['JavaScript', 'React', 'Node', 'MongoDB', 'Python'],
    'address': {
        'street': 'Space street',
        'zipcode': '02210'
    }
}
for key in person:
    if key == 'skills':
        for skill in person['skills']:
            print(skill)
```

For Else

If we want to execute some message when the loop ends, we use else.

```
# syntax
for iterator in range(start, end, step):
    do something
else:
    print('The loop ended')
```

Example:

```
for number in range(11):
    print(number)   # prints 0 to 10, not including 11
else:
    print('The loop stops at', number)
```

Pass

In python when statement is required (after semicolon), but we don't like to execute any code there, we can write the word *pass* to avoid errors. Also we can use it as a placeholder, for future statements.

Example:

```
for number in range(6):
    pass
```

🍪 You established a big milestone, you are unstoppable. Keep going! You have just completed day 10 challenges and you are 10 steps a head in to your way to greatness. Now do some exercises for your brain and muscles.

💻 Exercises: Day 10

Exercises: Level 1

1. Iterate 0 to 10 using for loop, do the same using while loop.
2. Iterate 10 to 0 using for loop, do the same using while loop.
3. Write a loop that makes seven calls to print(), so we get on the output the following triangle:

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

10. Use nested loops to create the following:

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

18. Print the following pattern:

```
0 x 0 = 0
1 x 1 = 1
2 x 2 = 4
3 x 3 = 9
4 x 4 = 16
5 x 5 = 25
6 x 6 = 36
```

```
7 x 7 = 49
8 x 8 = 64
9 x 9 = 81
10 x 10 = 100
```

29. Iterate through the list, ['Python', 'Numpy','Pandas','Django', 'Flask'] using a for loop and print out the items.

30. Use for loop to iterate from 0 to 100 and print only even numbers

31. Use for loop to iterate from 0 to 100 and print only odd numbers

Exercises: Level 2

1. Use for loop to iterate from 0 to 100 and print the sum of all numbers.

```
The sum of all numbers is 5050.
```

1. Use for loop to iterate from 0 to 100 and print the sum of all evens and the sum of all odds.

   ```
   The sum of all evens is 2550. And the sum of all odds is 2500.
   ```

Exercises: Level 3

1. Go to the data folder and use the countries.py file. Loop through the countries and extract all the countries containing the word *land*.
2. This is a fruit list, ['banana', 'orange', 'mango', 'lemon'] reverse the order using loop.
3. Go to the data folder and use the countries_data.py file.
   i. What are the total number of languages in the data
   ii. Find the ten most spoken languages from the data
   iii. Find the 10 most populated countries in the world

🎉 CONGRATULATIONS ! 🎉