Warm_up_06_loops_and_iterations

January 12, 2025

1 Dynamic Modelling Course - TEP4290: Warm-up 6

The point of this exercise is for you to practice what you have learned in the recommended videos and notebook on loops: - https://www.py4e.com/lessons/loops and - https://github.com/jakevdp/WhirlwindTourOfPython/blob/master/10-Iterators.ipynb

You will perform some basic operations that are designed to help you to get comforable with loops and iterations in Python. The exercise is pass/fail.

Good luck!

1.0.1 Quick summary

Infinite loops - while loops A while loop is a statement that will execute its body repeatetly as long as its condition is true: the following chunk of code will never stop (try if you want, you can just restart the kernel to stop it).

[1]:
```
#while True:
#    print('Infinite power!')
```

To make productive use of while loops we therefore often use an iteration variable - something that helps to stop it:

[2]:
```
bank_account = 1000
while bank_account >0:
    print('Go shopping!')
    bank_account -= 200
print('Go to work.')
```

```
Go shopping!
Go shopping!
Go shopping!
Go shopping!
Go shopping!
Go to work.
```

Another option is to use the break keyword:

[3]:
```
bank_account = 1000
while True:
    print('Go shopping!')
    bank_account -= 200
    if bank_account <0:
        print('Go to work.')
        break
```

```
Go shopping!
Go shopping!
Go shopping!
Go shopping!
Go shopping!
Go shopping!
Go to work.
```

Finite loops: Much more often you will need a finite loop, using a for statetment:

[4]:
```
liste = [1,2,3]
for element in liste:
    print(element)
```

```
1
2
3
```

For loops need to iterate over something - there are some nice tricks to this which you will pick up as you go (I recommend to just use google/stackoverflow to check for smart ways of doing what you want to do).

A simple example of this can be if you want to have both an element and its index from a list and can use enumerate:

[5]:
```
names = ['Superman', 'Batman', 'Green Lantern', 'Aquaman']
for index, name in enumerate(names):
    print(name, 'is on place', index)
```

```
Superman is on place 0
Batman is on place 1
Green Lantern is on place 2
Aquaman is on place 3
```

2 Tasks

Complete the tasks outlined below to achieve the same output as you find in the original file. Use the specific method described if applicable.

2.1 For loop print:

Using a for loop, write a function happy_wishes that prints happy new year for a list of three of your friends. Once you have wished all of them happy new year, print 'Done!'

[6]:
```python
friends = ['Superman', 'Batman', 'Green Lantern']
happy_wishes(friends)
```

```
Happy New Year: Superman
Happy New Year: Batman
Happy New Year: Green Lantern
Done!
```

2.2 Chessboard

A chessboard has rows that go from 1 to 8 and columns that go from A to H. Write a function using a for loop that returns all possibile positions a piece can be at, and puts them into a list.

[1]:
```python
def make_chessboard(columns, rows):
    chessboard = []
    for column in columns:
        for row in rows:
            position = f'{column}{row}'
            chessboard.append(position)
    return chessboard

# Starting point
columns = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H']
rows = range(1, 9)

# Generate the chessboard
my_board = make_chessboard(columns, rows)
print(my_board)
```

```
['A1', 'A2', 'A3', 'A4', 'A5', 'A6', 'A7', 'A8', 'B1', 'B2', 'B3', 'B4', 'B5',
'B6', 'B7', 'B8', 'C1', 'C2', 'C3', 'C4', 'C5', 'C6', 'C7', 'C8', 'D1', 'D2',
'D3', 'D4', 'D5', 'D6', 'D7', 'D8', 'E1', 'E2', 'E3', 'E4', 'E5', 'E6', 'E7',
'E8', 'F1', 'F2', 'F3', 'F4', 'F5', 'F6', 'F7', 'F8', 'G1', 'G2', 'G3', 'G4',
'G5', 'G6', 'G7', 'G8', 'H1', 'H2', 'H3', 'H4', 'H5', 'H6', 'H7', 'H8']
```

2.3 Finding things

Write a function that takes in a list of names and a single name and checks if that name occurs in the list using a for loop.

Hint: you can use break or just return (works like break if you want to terminate the entire function. How else could you solve that task?

[17]:
```python
def find(names_list, name_to_find):
    for name in names_list:
        if name == name_to_find:
            return True
    return False
```

```
list_of_names = ['Superman', 'Batman', 'Green Lantern', 'Batman', 'Aquaman',␣
 ↪'Batman']
name_to_find = 'Batman'
name_to_find2 = 'Flash'

find(['Superman', 'Batman', 'Green Lantern', 'Batman', 'Aquaman', 'Batman'],␣
 ↪'Batman')
find(['Superman', 'Batman', 'Green Lantern', 'Aquaman'], 'Flash')
find(list_of_names, name_to_find)
find(list_of_names, name_to_find2)
```

[17]:
```
False
```

[16]:
```
def find_name(names_list, name_to_find):
    return name_to_find in names_list

find_name(['Superman', 'Batman', 'Green Lantern', 'Batman', 'Aquaman',␣
 ↪'Batman'], 'Batman')
find_name(list_of_names, name_to_find)
```

[16]:
```
True
```

2.4 Counting things

Do the same as before: write a function that takes in a list of names and a single name, but this time return *how often* the name occurs in the list.

[18]:
```
def count(names_list, name_to_find):
    count = 0
    for name in names_list:
        if name == name_to_find:
            count += 1
    return count

count(['Superman', 'Batman', 'Green Lantern', 'Batman', 'Aquaman', 'Batman'],␣
 ↪'Batman')
```

[18]:
```
3
```

2.5 Fibonacci

Can you write a function called fibonacci that takes an integer n as an argument and returns the list of the n first numbers of the fibonacci sequence? Write the funtion and print the result for n=20.

A fibonaci sequence starts with two numbers (usually 0 and 1), each entry after that is the sum of the two prior. See also https://en.wikipedia.org/wiki/Fibonacci_number.

Hint: There are many solutions to this task - with varying efficiencies you might use for loops, while loops or recursion.

[19]:
```
def fibonnacci(n):
    """
    This function takes an integer n as an argument and
    returns the list of the n first numbers of the fibonacci list
    """
    if n <= 0:
        return []
    elif n == 1:
        return [0]
    elif n == 2:
        return [0, 1]

    fib_sequence = [0, 1]
    for i in range(2, n):
        next_number = fib_sequence[-1] + fib_sequence[-2]
        fib_sequence.append(next_number)

    return fib_sequence

print(fibonnacci(20))
```

```
[0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597, 2584,
4181]
```