Basic Techniques of Programming Part 2: Programming for Data Science with Python

5. Loops

Scenario: A Problem

Let's review the problem of calculating the diameter and circumference of a circle

```
It is assumed that a software developer is asked to write a Python
program that can calculate and print out the diameter and the
circumference of a circle. The user enters data of the radius and its
measurement unit (in, ft, cm, or m) from the console.
```

Let's write a Pseudo-Code:

1. Start
2. Read the input of the radius from the console
    - if (radius<0),
        - inform the user about the error
        - request to read again
3. Read the measurement unit of the radius (in, ft, cm, m)
    - if (unit is not among (in, ft, cm, m)),
        - inform the user about the error
        - request to read again
4. Calculate the diameter of the circle
    - diameter = 2 * radius
5. Calculate the circumference of the circle
    - Circumference + diameter * PI (3.14159)
6. Print out the diameter
7. Print out the circumference
8. End

Let's focus on this piece of pseudo-code:

Read the input of the radius from the console

- if (radius < 0),
  - inform the user about the error
  - request to read again

What happens if the user makes mistakes while entering the radius data again and again?

```
The program must perform the checking again and again until it can
read a valid piece of data._
```

In other words, the program must repeatedly check the input until it gets the correct one → the program uses LOOPS.

1. while Loop

Syntax

while (<loop-continuation condition.):

```
// ... ... ... statements(s)
```

Example:

radius = ... # radius of the circle

while (radius < 0):

```
print ("Radius cannot be negative!!!")

print ("Enter radius:")
```

radius = input("Enter radius: ")

IMPORTANT NOTES:

- In Python, WHILE loop also has an "ELSE" statement as IF does.
- However, it is strongly discouraged from using the ELSE statement of WHILE because it causes confusion and can make the code too complicated.
- For more details, see the example in the following section of FOR loop.

2. for Loop

The Python for loop is an iterator based for loop.

It steps through the items of lists, tuples, strings, the keys of dictionaries, and other iterables.

The Python for loop starts with the keyword for followed by an arbitrary variable name, which will hold the values of the following sequence object, which is stepped through.

Syntax

for (variable) in (sequence):

```
// ... ... ... statement(s)
```

Run the following code:

In [1]:
```
language = ["C", "C++", "Java", "Python", "Perl", "Ruby", "Scala"]
for x in language:
    print (x)
```

```
C
C++
Java
Python
Perl
Ruby
Scala
```

In [1]:
```
fav_Games = ["Elden Ring", "Baldur Gate 3", "Overwatch 1 & 2", "Mortal Kombat", "Stree
for games in fav_Games:
    print (games)
```

```
Elden Ring
Baldur Gate 3
Overwatch 1 & 2
Mortal Kombat
Street Fighter
Super Mario 64
```

IMPORTANT NOTES:

In Python, FOR loop also has an optional "ELSE" statement as IF statement does.

However, it is strongly discouraged from using the ELSE statement of FOR loop because it causes confusion and makes the code too complicated.

(Remember the Zen of Python: ... ***Simple is better than complex! Complex is better than complicated!***)

Semantically, the optional ELSE of FOR loop works exactly as the optional ELSE of a WHILE loop:

- It will be executed only if the loop hasn't been "broken" by a BREAK statement.
- So it will only be executed, after all the items of the sequence in the header have been interated through.

If a BREAK statement has to be executed in the program flow of the for loop:

- The loop will be exited
- The program flow will continue with the first statement following the FOR loop, if there is any at all.

Usually, BREAK statements are wrapped into conditional statements.

Run the following code:

In [2]:
```
edibles = ["ham", "Spam", "eggs", "nuts"]

for food in edibles:
    if food == "spam":
        print("No more spam please!")
        break
        print ("reat, delicious " + food)
    else:
        print("I am so glad: No spam!")
        print ("Finally, I finished stuffing myself")
```

```
I am so glad: No spam!
Finally, I finished stuffing myself
I am so glad: No spam!
Finally, I finished stuffing myself
I am so glad: No spam!
Finally, I finished stuffing myself
I am so glad: No spam!
Finally, I finished stuffing myself
```

In [4]:
```
fav_hobbies = ["study", "watch tv", "read", "play video games","sleep"]

for hobby in fav_hobbies:
    if hobby == "sleep":
        print("Snooze! One more wink wouldn't hurt.")
        break
    else:
        print("Yawn. I can't wait ", hobby)
        print("Hmm, I wonder what I should do...")
```

```
Yawn. I can't wait  study
Hmm, I wonder what I should do...
Yawn. I can't wait  watch tv
Hmm, I wonder what I should do...
Yawn. I can't wait  read
Hmm, I wonder what I should do...
Yawn. I can't wait  play video games
Hmm, I wonder what I should do...
Snooze! One more wink wouldn't hurt.
```

Let's consider the "else" statement in the FOR loop

for (variable) in (sequence):

```
    #statements ...
```

else:

```
    #statements ...
```

What does it mean by the "ELSE" statement?

Based on the syntax, it seems that the execution of the ELSE block is only based on the state of the conditional expression of the FOR statement - no other conditions.

However, semantically, it is not true!

The execution of the ELSE block only becomes meaningful due to the existence of a BREAK statement embedded inside another conditional statement like IF.

In the above piece of code, intuitively, the ELSE block would be executed if the conditional expression of FOR statement, i.e., "food is edibles", gets a "False" vaue - when the FOR loop has iterated through the whole sequence.

Let's execute the following piece of code that let the FOR loop iterate through its sequence:

Run the following code:

In [3]:

```
edibles = ["ham", "spam", "eggs", "nuts"]
for food in edibles:
    print("No break in FOR loop statements")
else:
    print ("I am so glad: No spam!")
print ("Finally, I finished stuffing myself")
```

```
No break in FOR loop statements
No break in FOR loop statements
No break in FOR loop statements
No break in FOR loop statements
I am so glad: No spam!
Finally, I finished stuffing myself
```

In [9]:

```
seasons = ["winter", "spring", "summer", "autumn"]
for season in seasons:
    if season == "winter":
        print("brrr")
    elif season == "spring":
        print("rain, rain, go away!")
    elif season == "summer":
        print("Who let the sun out?")
    elif season == "autumn":
        print("The only thing falling this season are the leaves.")
    else:
        print("Change is evitable.")
else:
    print("Is this else really necessary after all that?")
```

```
brrr
rain, rain, go away!
Who let the sun out?
The only thing falling this season are the leaves.
Is this else really necessary after all that?
```

NOTES: From the results of the above piece of code, the FOR loop iterated through the whole sequence. Therefore, the ELSE block is executed. However, the output of the ELSE block is not only meaningless but also misleading! There is "Spam" in the sequence! Let's find out what the developer really wants to achieve with the above ELSE statement - He/she wants to find out if "spam" is listed in the list of edibles by using FOR loop to iterate through the sequence. - If "spam" is found, he/she prints out a dialog to inform that. - Otherwise, he/she is so happy to print out that there is no "spam" in the list. Let's write another much simpler piece of code to achieve what he/she wants

Run the following 2 blocks of code:

In [4]:

```python
edibles = ["ham", "spam", "eggs","nuts"]
spam=False

for food in edibles:
    if food == "spam":
        spam = True
        print("No more spam please!")
        break
    print ("Great, delicious " + food)

if (not spam):
    print("I am so glad: No spam!")

print("Finally, I finished stuffing myself")
```

```
Great, delicious ham
No more spam please!
Finally, I finished stuffing myself
```

In [22]:

```python
groceries = ["Peanut Butter", "Bread", "Almond Milk", "Jelly", "Detergent", "Trash Bag
cereal = False

for grocery in groceries:
    if grocery == "Cereal":
        cereal == True
        print("Yay, you got cereal. Now, we need milk.")
        break
    print("I picked up ", grocery)

if (not cereal):
    print("How can you forget cereal?!")
print("Finally, I finished grocery shopping.")
```

```
I picked up  Peanut Butter
I picked up  Bread
I picked up  Almond Milk
I picked up  Jelly
I picked up  Detergent
I picked up  Trash Bags
Yay, you got cereal. Now, we need milk.
How can you forget cereal?!
Finally, I finished grocery shopping.
```

In [5]:

```
#NO spam

edibles =["ham","eggs","nuts"]
spam = False

for food in edibles:
    if food == "spam":
        spam = True
        print("No more spam please!")
        break
    print("Great, delicious " + food)
if (not spam):
        print("I am so glad: No spam!")
print("Finally, I finished stuffing myself")
```

```
Great, delicious ham
Great, delicious eggs
Great, delicious nuts
I am so glad: No spam!
Finally, I finished stuffing myself
```

In [28]:

```
groceries = ["Cereal"]
cereal = False

for grocery in groceries:
    if grocery == "Cereal":
        cereal == False
        print("All you got was cereal? We can't live on cereal.")
        break
    print("I picked up ", grocery)

shopping_list = ["Peanut Butter", "Bread", "Almond Milk", "Jelly", "Detergent", "Trash
for item in shopping_list:
    if (not cereal):
        print("How could you forget ", item, "?!")
    else:
        print("Finally, I finished grocery shopping.")
else:
    print("Next time, I am going shopping.")
```

```
All you got was cereal? We can't live on cereal.
How could you forget  Peanut Butter ?!
How could you forget  Bread ?!
How could you forget  Almond Milk ?!
How could you forget  Jelly ?!
How could you forget  Detergent ?!
How could you forget  Trash Bags ?!
Next time, I am going shopping.
```

IMPORTANT NOTES:

By not using the ELSE statement, the code is much easier to understand, much cleaner, and much simpler - following the Zen of Python:

> ***Simple is better than complex!***
>
> ***Complex is better than complicated!***

3. Break

"Break" is used to terminate a loop, i.e. completely get out of the loop, immediately.

Run the following 4 blocks of code:

In [6]:
```
for x in range (10,20):
       if (x == 15): break
       print (x)
```
```
10
11
12
13
14
```

In [29]:
```
for x in range (100,150,5):
       if (x == 135): break
       print (x)
```
```
100
105
110
115
120
125
130
```

In [7]:
```
for val in "string":
    if val == "i":
        break
    print(val)

print("The end")
```
```
s
t
r
The end
```

In [32]:
```
for val in "DeAundrie Howard":
    if val == "a":
        break
    print(val)

print("The end")
```

```
D
e
A
u
n
d
r
i
e
 
H
o
w
The end
```

In [8]:

```python
for letter in 'Python':
   if letter == 'h':
      break
   print ('Current Letter :', letter)
print ("Good bye!")
```

```
Current Letter : P
Current Letter : y
Current Letter : t
Good bye!
```

In [34]:

```python
for letter in 'DeAundrie Howard':
   if letter == 'a':
      break
   print ('Current Letter :', letter)
print ("Good bye!")
```

```
Current Letter : D
Current Letter : e
Current Letter : A
Current Letter : u
Current Letter : n
Current Letter : d
Current Letter : r
Current Letter : i
Current Letter : e
Current Letter :  
Current Letter : H
Current Letter : o
Current Letter : w
Good bye!
```

In [9]:

```python
# You will need to enter a price once you run the code.  Use high numbers, since the b
#  Hit return once you enter an amount

numItems=0
totalSales=0
totalSoldItems=5000

while(numItems<totalSoldItems):
    price=int(input("Enter the price of the next sold item: "))
    totalSales=totalSales+price
    if(totalSales>=1000000):
        break;
    numItems=numItems+1
print(totalSales)
```

```
Enter the price of the next sold item: 3000000000
3000000000
```

In [ ]:

```
# Enter intial current bank account balance followed by a projected number of weeks tc

curr_balance = float(input("Enter current savings balance:"))
savings_goal = float(input("Enter savings goal"))
mnthly_inc = float(input("Enter monthly income"))


while(curr_balance < savings_goal):
    wkly_income = float(input("Enter this weeks savings:"))
    savings = savings +500
    if (savings_goal >= dwn_pymt):
        break;
print(savings)
```

4. Continue

"Continue" is used to skip the rest of an itinerary and start a new one.

Run the following 2 blocks of code:

In [10]:

```
for val in "string":
    if val == "i":
        continue
    print(val)

print("The end")
```

```
s
t
r
n
g
The end
```

In [1]:

```
var = 10
while var > 0:
   var = var -1
   if var == 5:
      continue
   print ('Current variable value :', var)
print ("Good bye!")
```

```
Current variable value : 9
Current variable value : 8
Current variable value : 7
Current variable value : 6
Current variable value : 4
Current variable value : 3
Current variable value : 2
Current variable value : 1
Current variable value : 0
Good bye!
```

In [9]:

```
age = 15
while age > 0:
    age = age -1
    if age == 7:
        continue
    print("Happy", age, "birthday")
print("So...what's next?")
```

```
Happy 14 birthday
Happy 13 birthday
Happy 12 birthday
Happy 11 birthday
Happy 10 birthday
Happy 9 birthday
Happy 8 birthday
Happy 6 birthday
Happy 5 birthday
Happy 4 birthday
Happy 3 birthday
Happy 2 birthday
Happy 1 birthday
Happy 0 birthday
So...what's next?
```

In [ ]: