ClassPractice-Iterations

October 7, 2024

[ ]:

1 In Class Practice: Iterations

The goals of this lab are to help you to understand:

- How loops work.
- The difference between definite and indefinite loops, and when to use each.
- How to build an indefinite loop with complex exit conditions.
- How to create a program from a complex idea.

2 Understanding Iterations

Iterations permit us to repeat code until a Boolean expression is `False`. Iterations or **loops** allow us to write succinct, compact code. Here's an example, which counts to 3 before Blitzing the Quarterback in backyard American Football:

[ ]:
```
i = 1
while i <= 3:
    print(i,"Mississippi...")
    i=i+1
print("Blitz!")
```

2.1 Breaking it down...

The `while` statement on line 2 starts the loop. The code indented beneath the `while` (lines 3-4) will repeat, in a linear fashion until the Boolean expression on line 2 `i <= 3` is `False`, at which time the program continues with line 5.

2.1.1 Some Terminology

We call `i <=3` the loop's **exit condition**. The variable `i` inside the exit condition is the only thing that we can change to make the exit condition `False`, therefore it is the **loop control variable**. On line 4 we change the loop control variable by adding one to it, this is called an **increment**.

Furthermore, we know how many times this loop will execute before it actually runs: 3. Even if we allowed the user to enter a number, and looped that many times, we would still know. We call this

a **definite loop**. Whenever we iterate over a fixed number of values, regardless of whether those values are determined at run-time or not, we're using a definite loop.

If the loop control variable never forces the exit condition to be `False`, we have an **infinite loop**. As the name implies, an Infinite loop never ends and typically causes our computer to crash or lock up.

2.1.2 For loops

To prevent an infinite loop when the loop is definite, we use the `for` statement. Here's the same program using `for`:

[ ]:
```
for i in range(1,4):
    print(i,"Mississippi...")
print("Blitz!")
```

One confusing aspect of this loop is `range(1,4)` why does this loop from 1 to 3? Why not 1 to 4? Well it has to do with the fact that computers start counting at zero. The easier way to understand it is if you subtract the two numbers you get the number of times it will loop. So for example, 4-1 == 3.

2.1.3 1.1 You Code

In the space below, Re-Write the above program to count Mississippi from 10 to 15. You need practice **writing** loops, so make sure you do NOT copy the code.

Note: How many times will that loop?

[ ]:
```
# TODO Write code here
```

2.2 Indefinite loops

With **indefinite loops** we do not know how many times the program will execute. This is typically based on user action, and therefore our loop is subject to the whims of whoever interacts with it. Most applications like spreadsheets, photo editors, and games use indefinite loops. They'll run on your computer, seemingly forever, until you choose to quit the application.

The classic indefinite loop pattern involves getting input from the user inside the loop. We then inspect the input and based on that input we might exit the loop. Here's an example:

[ ]:
```
i = 1
while True:
    print(i, "mississippi")
    if i == 10:
        break
```

[ ]:
```
name = "pedro"
while True:
    guess = input("Guess my name: ")
    if name == guess:
```

```
        print("You guessed my name")
        break
    else:
        print(f"My name is not {guess}")
```

In the above example, the loop will keep on looping until we enter `mike`. The value `mike` is called the **sentinal** value - a value we look out for, and when it exists we stop the loop. For this reason indefinite loops are also known as **sentinal-controlled loops**.

The classic problem with indefinite/sentinal controlled loops is that its really difficult to get the application's logic to line up with the exit condition. For example we need to set `name = ""` in line 1 so that line 2 starts out as `True`. Also we have this wonky logic where when we say `'mike'` it still prints `Nope, my name is not mike!` before exiting.

2.2.1 Break statement

The solution to this problem is to use the break statement. **break** tells Python to exit the loop immediately. We then re-structure all of our indefinite loops to look like this:

```
while True:
    if sentinel-controlled-exit-condition:
        break
```

Here's our program we-written with the break statement. **This is the recommended way to write indefinite loops in this course.**

NOTE: We always check for the sentinal value immediately AFTER the `input()` function.

[ ]:
```
while True:
    name = input("Say my name!: ")
    if name == 'mike':
        break
    print("Nope, my name is not %s!" %(name))
```

2.2.2 1.2 You Code: Debug This loop

This program should count the number of times you input the value `ni`. As soon as you enter a value other than `ni` the program will stop looping and print the count of `ni`'s.

Example Run:

```
What say you? ni
What say you? ni
What say you? ni
What say you? nay
You said 'ni' 3 times.
```

The problem of course, is this code wasn't written correctly. Its up to you to get it working!

[ ]:
```
#TODO Debug this code
nicount=0
```

```
while True:
    say = input "What say you? ")
    if say == 'ni':
        break
    nicount = 1
print(f"You said 'ni' P {nicount} times.")
```

2.3 Multiple exit conditions

This indefinite loop pattern makes it easy to add additional exit conditions. For example, here's the program again, but it now stops when you say my name or type in 3 wrong names.

Make sure to run this program a couple of times to understand what is happening:

- First enter mike to exit the program,
- Next enter the wrong name 3 times.

[ ]:
```
times = 0
while True:
    name = input("Say my name!: ")
    times = times + 1
    if name == 'mike': # sentinal 1
        print("You got it!")
        break
    if times == 3: # sentinal 2
        print("Game over. Too many tries!")
        break
    print(f"Nope, my name is not {name}")
```

3 Counting Characters in Text

Let's conclude with you writing your own program that uses both definite and indefinite loops. This program should input some text and then a character, counting the number of characters in the text. This process will repeat until the text entered is empty.

The program should work as follows. Example run:

```
Enter a text, or press ENTER quit: mississippi
Which character are you searching for? i
There are 4 i's in mississippi

Enter a text, or press ENTER quit: port-au-prince
Which character are you searching for? -
There are 4 -'s in port-au-prince

Enter a text, or press ENTER quit:
Goodbye!
```

This seems complicated, so let's break the problem up using the **problem simplification** approach.

First write code to count the numbers of characters in any text. Here is the algorithm:

```
set count to 0
input the text
input the search character
for ch in text
    if ch equals the search character
        increment the count
print there are {count} {search characters} in {text}
```

3.0.1 1.3 You Code

Implement the algorithm above in code in the cell below.

[ ]:
```
# TODO Write code here
```

Next, we surround the code we wrote in 1.4 with a sentinal-controlled indefinite loop. The sentinal (the part that exits the loop is when the text is empty (`text==""`) The algorithm is:

```
loop
    set count to 0
    input the text
    if text is empty quit loop
    input the search character
    for ch in text
        if ch equals the search character
            increment the count
    print there are {count} {search characters} in {text}
```

3.0.2 1.4 You Code

Implement the algorithm above in code.

[ ]:
```
# TODO Write Code here:
```

4 Metacognition

4.0.1 Rate your comfort level with this week's material so far.

1 ==> I don't understand this at all yet and need extra help. If you choose this please try to articulate that which you do not understand to the best of your ability in the questions and comments section below.
2 ==> I can do this with help or guidance from other people or resources. If you choose this level, please indicate HOW this person helped you in the questions and comments section below.
3 ==> I can do this on my own without any help.
4 ==> I can do this on my own and can explain/teach how to do it to others.

`--== Double-Click Here then Enter a Number 1 through 4 Below This Line ==--`