Class 5-6 Loops

Tuesday, February 13, 2024 9:18 AM

lecture 3

- Please, put cellphones away from your desk
- Feel free to interrupt with a question
- Raise hand to respond to a question

- Practice - problems from Handout 3
- Quiz on Feb 23 - **guide** and practice are posted
  - To prepare - review reading, do practice problems, work on homework assignment

- Do you have any questions?

Handout 3

Repetition and loop constructs

iteration

Loops are used for implementing repetition. Python loop statements:

- while
- for

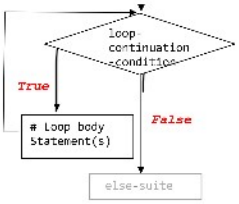
One repetition, i.e. one pass through the loop, i.e. one execution of the loop body is called **an iteration.**

WHILE LOOP

```
while loop-continuation-condition:
    # Loop body
    Statement(s)
```

Two common loop types:

- counting loops - know exactly how many times to repeat a set of actions
  a. usually done with a use of a counter variable
  b. counter is initialized before the loop starts executing
  c. counter is updated after each iteration of the loop



- conditional loops – can't predict how many times will repeat, but know at what condition to stop.

Examples: notice the indentation:

1.
```
i = 0  # the counter variable - to keep track of
       # number of times gone through the loop
m = 5;
while i < m:
    print('i is', i , 'now.')
    i += 1
print("Done. i is " , i);
```

i 0 1 . . . . .

m 5

i is 0 now.
i is 1 now
i is 2 now
...
i is 4 now
Done, i is 5

Question: What will happen if m = -5? If the `i -=1;` in the loop body is omitted?

Programming and Debugging Pitfalls: infinite loops, off by one

2.

```
# Using while repetition, generate a table of
# CD interest rates and interest earned

invest = float(input('Enter initial investment ($): '))
years = int(input('Enter years: '))

rateMin = float(input('Minimum rate: '))
rateMax = float(input('Maximum rate: '))
rateIncr = float(input('Increment rate: '))

 # print the header rows for the table
print()
print('Interest', 'Interest')
print('  Rate  ', '  Earned')

 # generate a table of interest rates and corresponding interest
earned
rate = rateMin # the initial rate for the table
while rate <= rateMax:
    amount = invest * (1 + rate / 100) ** years
    interest = round(amount - invest, 2)

    print(format(rate, '6.3f'), format(interest, '8,.2f'))

    rate += rateIncr # the next rate
```

```
Enter initial investment ($): 1000
Enter years: 3
Minimum rate: 2
Maximum rate: 3
Increment rate: .25

Interest Interest
    Rate   Earned
   2.000    61.21
   2.250    69.03
   2.500    76.89
   2.750    84.79
   3.000    92.73
```

Questions:

1. Which variables affect the number of rows in the table?
2. What about the number of columns?
3. How would you modify the code above to print interest earned in the intermediate years (year 1, year 2, year 3 ...) as well?

PRACTICE PROBLEMS ON LOOPS

1. Read input until user enters a vowel.
2. Write a code segment that gives the user 5 attempts at guessing a number between 0 and 32. Stop if a number was guessed correctly or 5 attempts have been made.
   Hint: To produce a random integer within [0,32) range, include the following code:
   ```
   import random #put in the top of the program
   num = random.randint(0, 32)
   ```
3. Read numbers until user enters a negative number. Display how many numbers among those entered were multiples of 3.
4. Write a code segment that keeps reading words entered in all lowercase letters, so long they appear in alphabetical order. Once the order is violated, stop and print out how many words were entered before the order was violated.
5. Read numbers representing a sequence of daily values of a stock over some number of days. For each new number calculate the difference from the previous day and stop after observing 5 increases in value.
6. Read numbers representing a sequence of daily values of a stock over some number of days, until user enters a negative number. For each new number, starting from the second one, output the percentage of value change from the previous day.

FOR LOOP

```
for var in sequence-producing-expression:
    # Loop body
    Statement(s)
```

The for loop will execute the Statement(s) in the loop body for each value of variable var. var will take on values from the sequence, in turn.

Notes:

- the sequence-producing-expression is evaluated **once** before the loop's first iteration.
- the loop body is run **for each element in the sequence in turn,** from first to last
- var is updated for each iteration to store the next value of the sequence, from first to last.

Examples: what will be printed?

```
for var in range(1,10):
    print(var, end = " ")

for ch in "bentley":
    print(ch)

for var in [1, 56, 67]:
    print(var, end = " ")
```

Equivalent to the following:
(assuming i, seq are new variables)

```
seq = sequence-producing-expression
if len(seq) > 0:
    i = 0
    while i < len(seq):
        var = seq[i]
        # Loop body
        Statement(s)
        i+=1
```

RANGE FUNCTION

range([*start*,] *stop*[, *step*]) - is a constructor (i.e. a function that creates an object) Often used for counting loops. Returns a **range** object (that looks like a list of numbers)
The arguments *start*, *stop*, *step* must be integers. If the step argument is omitted, it defaults to 1. If the start argument is omitted, it defaults to 0.

Examples:

```
>>> list(range(10))
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> list(range(1, 11))
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
>>> list(range(0, 10, 3))
[0, 3, 6, 9]
>>> list(range(0, -10, -1))
[0, -1, -2, -3, -4, -5, -6, -7, -8, -9]
>>> list(range(0))
[]
>>> list(range(1, 10, 0))
ValueError: range() step argument must not be zero
```

PRACTICE PROBLEMS ON LOOPS:

7. Print all Olympic years in the 21st century, including a designation of which games (winter or summer) will be conducted.
8. Ask user how many times to generate a random number and run a loop generating that many random integers between 10 and 25, then output how many of them were even.
9. Print a multiplication table as shown (no coloring required)

Multiplication

X	1	2	3	4	5	6	7	8	9	10	11	12
1	1	2	3	4	5	6	7	8	9	10	11	12
2	2	4	6	8	10	12	14	16	18	20	22	24
3	3	6	9	12	15	18	21	24	27	30	33	36
4	4	8	12	16	20	24	28	32	36	40	44	48
5	5	10	15	20	25	30	35	40	45	50	55	60
6	6	12	18	24	30	36	42	48	54	60	66	72
7	7	14	21	28	35	42	49	56	63	70	77	84
8	8	16	24	32	40	48	56	64	72	80	88	96
9	9	18	27	36	45	54	63	72	81	90	99	108
10	10	20	30	40	50	60	70	80	90	100	110	120
11	11	22	33	44	55	66	77	88	99	110	121	132
12	12	24	36	48	60	72	84	96	108	120	132	144

10. Ask user to enter a string, calculate how many vowels there are in that string.