Handout 4

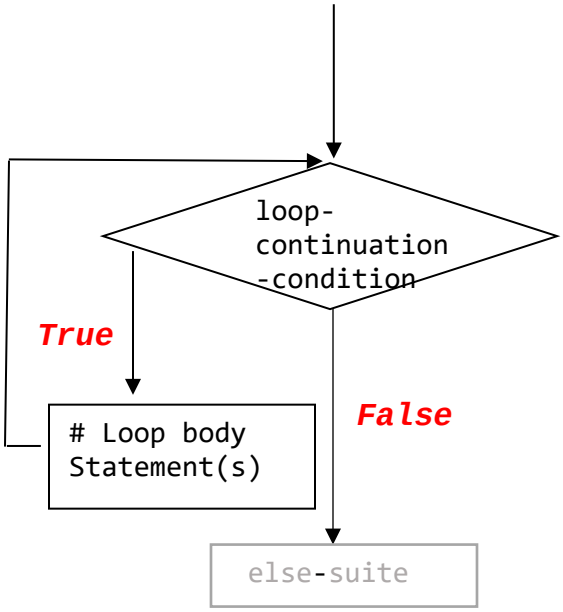
Repetition and loop constructs.

Loops are used for implementing repetition. Python loop statements:

while
for

WHILE LOOP

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



Several logical loop organizations

1. counting loops

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

2. conditional loops

– continue while the continuation condition is true

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

GENERAL FORM

(from https://docs.python.org/3/reference/)

```
while_stmt ::=  "while" expression ":" loop-body-suite
                ["else" ":" else-suite]
```

This repeatedly tests the expression and, if it is true, executes the first suite (`loop-body-suite`); if the expression is false the suite of the `else` clause, if present, is executed and the loop terminates.

A `break` statement executed in the `loop-body-suite` terminates the loop without executing the `else` clause's suite. A `continue` statement executed in the `loop-body-suite` skips the rest of the suite and goes back to testing the expression.

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);
```

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

PRACTICE PROBLEMS ON LOOPS

1. Write a code segment that prints out all characters of a string one character per line in reverse, from last one to the first one.
2. Write a code segment that allows the user to enter 10 numbers and computes and prints out their sum, average, product.
3. Read input until user enters an even number.
4. Write a code segment that gives the user 5 attempts at entering a word with at least 7 characters. Stops the loop after the entered word is 7 chars or longer, or after 5 attempts were exhausted.
5. Write a code segment that keeps reading words, so long that they appear in alphabetical order. Once the order is violated, stop and print out how many words started with a vowel.

FOR LOOP

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

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.

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
```

Examples: what will be printed?

```
for ch in "bentley":
    print(ch)
```

```
str = "bentley"
for ch in str:
    str = 'foo'
    print(ch)

str = input("Please enter a sentence \n")
for word in str.split():
    if (len(word) > 3) :
        print(word)


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

RANGE

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), as follows.

From https://docs.python.org/3/library/stdtypes.html:

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, 30, 5))
[0, 5, 10, 15, 20, 25]
>>> 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, 0))
[]
>>> list(range(1, 10, 0))
ValueError: range() step argument must not be zero
```

The arguments to the range constructor must be integers. If the *step* argument is omitted, it defaults to 1. If the *start* argument is omitted, it defaults to 0. If *step* is zero, ValueError is raised.

For a positive *step*, the contents of a range r are determined by the formula r[i] = start +step*i where i >= 0 and r[i] < stop.

For a negative *step*, the contents of the range are still determined by the formula r[i] = start+ step*i, but the constraints are i >= 0 and r[i] > stop.

The advantage of the range type over a regular list or tuple is that a range object will always take the same (small) amount of memory, no matter the size of the range it represents (as it only stores the start, stop and step values, calculating individual items and subranges as needed).

PRACTICE PROBLEMS ON LOOPS:

6. Print every second word in a sentence inputted by the user.
7. Print all Olympic years in the 21st century, including a designation of which games (winter or summer) will be conducted.
8. Print the following table: each row from (1 to 25) must contain numbers in the range between 0 and 25, divisible by the row number.

GENERAL FORM

```
for_stmt ::= "for" target_list "in" expression_list ":" suite
             ["else" ":" suite]
```

BREAK AND CONTINUE STATEMENTS

SHOULD BE AVOIDED!!!

break – stop looping, go to the next statement after the loop.
continue – stop the current iteration of the loop, go to the condition again

PRACTICE PROBLEM ON LOOPS:

9. Write a program that plays a game of Hangman. Hangman is a two-player game in which one player, in our case, the program, picks a word (you can hardcode it) that the other player (the user) has to guess. The program originally reveals only the first and the last letters, and dashes for every other letter, e.g. `N------K`, if the word is `NOTEBOOK`. At each turn of the game, the user suggests a letter and the program responds by either stating that the word does not have the letter, or revealing the letter in the word, e.g. `O` in `NO---OOK`.

   The game stops when either
   1. user guesses the entire word, and then user is the winner, or
   2. when user has proposed 7 letters not found in the word. In this case, the program wins.

Here's a sample run of the program:

```
Let's play a game of Hangman.
I have picked a word that starts with letter T and ends with E.
Here's the template in which each dash denotes a single letter:
T - - - - - - - - - E
Now it's your turn to guess a letter.

Please enter your next guess: F
Incorrect. Letter F does not occur in this word.

Please enter your next guess: E
Correct.
T E - - E - - - - - E

Please enter your next guess: Z
Incorrect. Letter Z does not occur in this word.

Please enter your next guess: P
Correct.
T E - P E - - - - - E

Please enter your next guess: Q
Incorrect. Letter Q does not occur in this word.

Please enter your next guess: O
Incorrect. Letter O does not occur in this word.

Please enter your next guess: L
Incorrect. Letter L does not occur in this word.

Please enter your next guess: I
Incorrect. Letter I does not occur in this word.

You lost this game. The word is TEMPERATURE
```