Module 03: Flow control

Agenda

- Relational and Logical Operators
- If statement
- While loop
- For loop
- Loop Control Statements

Relational and Logical Operators

- Logical Operators:
  - and - Logical AND
  - or - Logical OR
  - not - Logical NO
- Two sets of relational operators:

Example Expression	Relational operator	Symbol
result = val2 == val3	Is val2 equal to val3?	==
result = val2 != val3	Is val2 not equal to val3?	!=
result = val2 > val3	Is val2 greater than val3?	>
result = val2 >= val3	Is val2 greater than or equal to val3?	>=
result = val2 < val3	Is val2 smaller than val3?	<
result = val2 <= val3	Is val2 smaller than or equal to val3?	<=

If statment

if statement1:

…

elif statement2:

……..

else:

……

- *elif* and *else* statements are optional

- Example:

```
number = 10
if number %2 == 0:
    print ("The number is even")
else:
    print ("The number is odd")
```

While loop

- Python while loops are used for repeating sections of code - but unlike a for loop, the while loop will run as long as defined condition is met.
- while expression:
  statement(s)

Example 1:

```
count = 0
while  count < 9:
   print ('The count is:', count)
   count += 1
```

Example 2:

```
n = input("Please enter 'hello':")
while n.strip() != 'hello':
        n = input("Please enter
'hello':")
```

For Loop

- Python for loops iterates over the member of a sequence in order

Example 1:

```
for letter in 'sentence':
   print (letter)       #prints: s e n t e n c e
```

Example 2:

```
for num in range(10,20):       #prints 10, 11, 12 …19
   print(num)
```

Example 3:

```
for num in range(10,21, 2):     #prints 10, 12, 14 …20
   print(num)
```

Loop Control Statements

- break

  The **break** statement in Python terminates the current loop and resumes execution at the next statement, just like the traditional break found in C.

- continue

  The **continue** statement in Python returns the control to the beginning of the while loop.

- pass

  The **pass** statement in Python is used when a statement is required syntactically but you do not want any command or code to execute.

  The **pass** statement is a *null* operation; nothing happens when it executes. The **pass** is also useful in places where your code will eventually go, but has not been written yet

Loop Control Statements – cont’d

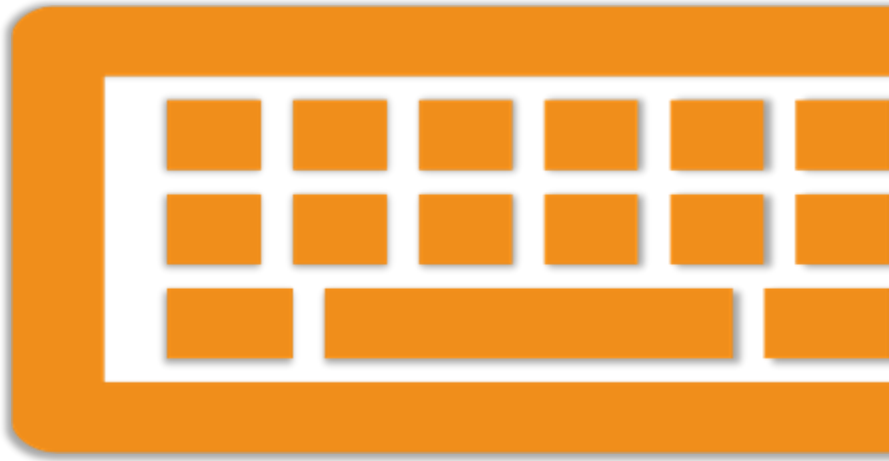
```
import math

i = 5
print("I will print the square root of 5 non-negative numbers (0 to exit)")

while i:
    s = input("Enter the next number => ")
    num = int(s)
    if num == 0:  # user wants to exit
            break
        if num < 0:
            continue
    i -= 1
    print(math.sqrt(num))
```

Input Check

Demo

Conslution Labs 01-08

Lab



Questions