5zgauzszo

September 28, 2023

**#CSC1001 TUTORIAL 4 - FLOW CONTROL**

Frederick Khasanto - 122040014 - 28 Sept 2023

# 1 CONDITIONAL FLOW

## 1.1 Comparison expressions

returns `True` or `False`

`a < b` : a less than b

`a > b` : a more than b

`a <= b` : a less than or equal to b

`a >= b` : a more than or equal to b

`a == b` : a is equal to b (Note: `==` is used for comparison, while `=` is used for value assignment)

`a != b` : a is not equal to b

**Comparing numbers**

[ ]:
```
2 < 1  # False
```

[ ]:
```
False
```

[ ]:
```
7 == 7.0  # True
```

[ ]:
```
True
```

[ ]:
```
1 == 10*0.1 # True
```

[ ]:
```
True
```

[ ]:
```
1 == 0.1+0.1+0.1+0.1+0.1+0.1+0.1+0.1+0.1+0.1   # False (1 != 0.999999...)␣
↪precision loss
```

[ ]:
```
False
```

**Comparing strings** (according to ASCII values, see https://www.ascii-code.com/ DEC-Symbol columns)

[ ]:
```
'a' < 'A'  # False
```

[ ]: False

[ ]:
```
'2' < 'a'  # True
```

[ ]: True

[ ]:
```
'aaa' < 'aab'  # True
```

[ ]: True

[ ]:
```
'abcd' < 'aaAA'  # False
```

[ ]: False

[ ]:
```
'A' == 65  # False; cannot directly compare, should be ord('A') == 65 => True
```

[ ]: False

[ ]:
```
ord('A') == 65 # True
```

[ ]: True

## 1.2 Logical Operators

not, and, or

[ ]:
```
not True
```

[ ]: False

[ ]:
```
print(True and True)
print(True and False)
print(False and True)
print(False and False)
```

```
True
False
False
False
```

[ ]:
```
print(True or True)
print(True or False)
print(False or True)
```

```
print(False or False)
```

```
True
True
True
False
```

### 1.3 Conditional statements

```
if (condition 1):
  ...
elif (condition 2):
  ...
elif (condition 3):
  ...
else:
  ...
```

[ ]:
```
x = 2
# One-way decision (if)
if (x > 1 and x < 5):   # 1 < x < 5
    print("(one-way) run if block")
print("Always runs")
```

```
(one-way) run if block
Always runs
```

[ ]:
```
# Two-way decision (if-else)
if (x > 2):
    print("(two-way) run if block")
else:
    print("(two-way) run else block")
print("Always runs")
```

```
(two-way) run else block
Always runs
```

[ ]:
```
# Multi-way decision (if-elif-...-else)
if (x < 5):   # 1st condition
    print("(multi-way) run if block")
elif (x < 4): # 2nd condition
    print("(multi-way) run 1st elif block")
elif (x < 3): # 3rd condition
    print("(multi-way) run 2nd elif block")
else:
    print("(multi-way) run else block")
print("Always runs")
```

```
(multi-way) run if block
Always runs
```

# 2 REPEATED FLOW

## 2.1 Range

`range(start, stop, step)`

–start: (optional) starting number (default 0)

–stop: (required) stopping number (excluded)

–step: (optional) incrementation (default 1)

[ ]:
```
range(5)
```

[ ]:
```
range(0, 5)
```

[ ]:
```
type(range(5))
```

[ ]:
```
range
```

[ ]:
```
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(10,0,-1))
```

[ ]:
```
[10, 9, 8, 7, 6, 5, 4, 3, 2, 1]
```

[ ]:
```
list(range(20,10,-2))
```

[ ]:
```
[20, 18, 16, 14, 12]
```

[ ]:
```
list(range(10,1,1))
```

[ ]:
```
[]
```

[ ]:
```
list(range(1,10,-1))
```

[ ]: []

### 2.2 in keyword

[ ]:
```
friends = ['A','B','C']
for friend in friends:
  print('Hi,', friend)
print('Done')
```

```
Hi, A
Hi, B
Hi, C
Done
```

[ ]:
```
for c in 'csc1001':
    print(c)
```

```
c
s
c
1
0
0
1
```

### 2.3 for Loop

- definite: execute on exact number of times
- no. of iterations = no. of elements in the set

[ ]:
```
for i in range(0, 10, 2):
    print(i)
```

```
0
2
4
6
8
```

### 2.4 while Loop

- indefinite: keeps going until the logical condition becomes `False`

```
# forever loop
while True:
  ...
```

[ ]:
```
n = 5
while n > 0:
```

```
  print(n)
  n -= 1
```

```
5
4
3
2
1
```

### 2.5 break & continue

`break`: terminates the loop

`continue`: directly skip to the next iteration

[ ]:
```
for i in range(5):
  if i == 2:
    break  # stops when i = 2 and directly get out of the loop
  print(i)
print('done')
```

```
0
1
done
```

[ ]:
```
i = 0
while i < 5:
  if i == 2:
    break
  print(i)
  i += 1
print('done')
```

```
0
1
done
```

[ ]:
```
for i in range(5):
  if i == 2:
    continue  # skips number 2, continue to 3
  print(i)
print('done')
```

```
0
1
3
4
done
```

[ ]:
```
i = 0
while i < 5:
  if i == 2:
    i += 1  # Update the i value before continuing
    continue
  print(i)
  i += 1
print('done')
```

```
0
1
3
4
done
```

### 2.6 `try` and `except`

- To handle errors in the code
- If the code in the `try` block works, the `except` block will be skipped.
- If the code in the `try` block fails (produce errors), the `except` block will be executed.

[ ]:
```
str1 = "Hello A"
try:
  x = int(str1)  # ValueError
except:
  x = -1  # This will be run
  print("There is an error.")
print(x)
```

```
-1
```

[ ]:
```
str2 = "123"
try:
  x = int(str2)  # No error, code will be run
except:
  x = -1
  print("There is an error.")
print(x)
```

```
123
```

[ ]:
```
try:
  num = int(input("Please enter a number: "))  # ValueError if user does not␣
 ↪input numbers
  print("You inputted", num)
except:
  print("Your input is not a number!")
```

```
Please enter a number: abcd
Your input is not a number!
```

[ ]:
```python
# Keep asking for input until user gives a correct input
while True:
  try:
    num = int(input("Please enter a number: "))
    print("You inputted", num)
    break
  except:
    print("Your input is not a number!")
```

```
Please enter a number: abcd
Your input is not a number!
Please enter a number: csc1001
Your input is not a number!
Please enter a number: 20
You inputted 20
```

## 3 Practice Questions

##Q1: Quadratic equation

[ ]:
```python
##Prompt the users to input the coefficients
a,b,c=eval(input("Enter coefficients a,b and c in the equation ax^2+bx+c=0:"))

##Calculate the discriminant (= b^2-4ac)
discriminant=b**2-4*a*c

##Obtain the roots based on different conditions for the discriminant
if discriminant>0:
    x1=(-b+discriminant**0.5)/2/a
    x2=(-b-discriminant**0.5)/2/a
    print('The two roots of the equation are x1=%.2f and x2=%.2f.'%(x1,x2))
elif discriminant==0:
    x=-b/2/a
    print("There is only one root x=%.2f."%x)
else:
    print("The equation has no real roots.")
```

##Q2: Days in a month

[ ]:
```python
##Prompt the user to enter the month and year
month,year=eval(input("Enter the month and year:"))

##Use multi-way decision flow to handle different cases for different month
if month==1:
##  Use numOfDay to store the number of days
```

```
    numOfDay=31
    print("January %d has %d days"%(year,numOfDay))
elif month==2:
    ##  Conditions for leap year( )
    if (year%4==0 and year%100 != 0) or year%400==0:
        numOfDay=29
        print("February %d has %d days"%(year,numOfDay))
    else:
        numOfDay=28
        print("February %d has %d days"%(year,numOfDay))
elif month==3:
    numOfDay=31
    print("March %d has %d days"%(year,numOfDay))
elif month==4:
    numOfDay=30
    print("April %d has %d days"%(year,numOfDay))
elif month==5:
    numOfDay=31
    print("May %d has %d days"%(year,numOfDay))
elif month==6:
    numOfDay=30
    print("June %d has %d days"%(year,numOfDay))
elif month==7:
    numOfDay=31
    print("July %d has %d days"%(year,numOfDay))
elif month==8:
    numOfDay=31
    print("August %d has %d days"%(year,numOfDay))
elif month==9:
    numOfDay=30
    print("September %d has %d days"%(year,numOfDay))
elif month==10:
    numOfDay=31
    print("October %d has %d days"%(year,numOfDay))
elif month==11:
    numOfDay=30
    print("November %d has %d days"%(year,numOfDay))
else:
    numOfDay=31
    print("December %d has %d days"%(year,numOfDay))
```

##Q3: Sum the digits

[ ]:
```
while True:
    ##  Prompt up the indicator for user to enter
    number = eval(input("Enter a number:"))
    ##  Use sumup to store the sum of digits
```

```
    sumup = 0
    ##  if number is bigger than 0, there are digits to be summed, so go into␣
 ↪the loop
    while number > 0:
        ##  sum up the digit
        sumup += number%10
        ##  remove the digit already being summed
        number //= 10

    ##  Display the result
    print("Sum up all the digits as:", sumup)

    ##  Use decision to store the string determining whether to continue or not
    decision = input("Do you want to continue? y or n:")
    ##  If you input 'y', then continue to enter a number, otherwise stop
    if decision !='y':
        break
    ##  This part can be omitted
    else:
        continue
```

##Q4: Count numbers

[ ]:
```
##Give an initial value other than 0
integer = None
##To count positive numbers
countPositive = 0
##To count negative numbers
countNegative = 0
##To count total numbers
count = 0
##To sum up the numbers
sumOfNum = 0

##if integer==0, stop the loop
while integer != 0:
    ##  Use try except to capture the error input
    try:
        integer = int(input("Enter an integer, the input ends if it is 0: "))
    except:
        print("Invalid number! Please input again!")
        ##  Continue the loop while not execute the following code
        continue

    if integer > 0:
        countPositive += 1
    elif integer < 0:
```

```
        countNegative += 1
    else:
        ##  Continue without executing the following code, in this case␣
 ↪integer=0, will break the loop
        continue

    count += 1
    sumOfNum += integer

if count==0:
    print("No numbers are input except 0")
else:
    print("The number of positives is %d"%countPositive)
    print("The number of negatives is %d"%countNegative)
    print("The sum of numbers is %d"%sumOfNum)
    print("The average of numbers is %.2f"%(sumOfNum/count))
```

##Q5: Find factors

[ ]:
```
##Use try except to capture the error of not entering a number
try:
    integer=int(input("Enter an integer:"))
except:
    integer=None

if integer is None:
    print("The input is invalid!")
else:
    ##  Start from 2
    factor=2
    ##  Iterate from 2 to the integer
    while factor<=integer:
        if integer%factor==0:
            ##  Stop when factor equal to the integer
            if integer==factor:
                print(factor)
                break
            print(factor,end=',')
            integer/=factor
        else:
            factor+=1
```

##Q6: Display a pyramid

[ ]:
```
##Input the number of lines of the pyramid
try:
    numOfLine=int(input("Enter the number of lines:"))
```

```
except:
    numOfLine=None

if numOfLine is None:
    print("The input is invalid!")
else:
    ##Diplay each row of the pyramid
    for row in range(1,numOfLine+1):

        ##  Display the left part of the row
        for column in range(numOfLine,1,-1):
            if column<=row:
                ##  Print the number
                print("%4d"%column,end='')
            else:
                ##  Print space
                print("%4s"%" ",end='')

        ##  Display the right part of the row
        for column in range(1,numOfLine+1):
            if column<=row:
                ##  Print the number
                print("%4d"%column,end='')
            else:
                ##  Print space
                print("%4s"%" ",end='')

        ##  Change a new line to print
        print()
```