6 Repetition Control Instruction

"Merry go round..."

Contents

KanNotes

Repetition Control Instruction

- It helps us a repeat a set of statements in a program. There are two types of repetition control instructions:

  (a) while
  (b) for

  Unlike many other languages there is no do-while loop in Python.

- **while** is used to repeatedly execute instructions as long as condition is true. It has two forms:

```
while condition :
    statement1
    statement2
```

```
while condition :
    statement1
    statement2
else :
    statement3
    statement4
```

  - **else** block is optional. If present, it is executed when **condition** fails.
  - If the **while** loop is terminated abruptly using a **break** statement then the **else** block is not executed.

- **for** is used to iterate over elements of a sequence such as string, tuple or list. It has two forms:

```
for var in list :
    statement1
    statement2
```

```
for var in list :
    statement1
    statement2
else :
    statement3
    statement4
```

  - During each iteration **var** is assigned the next value from the list.
  - In place of a list a string, tuple, set or dictionary can also be used.
  - **else** block is optional. If present, it is executed if loop is not terminated abruptly using **break**.

Usage of *while* loop

- A **while** loop can be used in following three situations:
  - Repeat a set of statements till a condition remains True.

- Repeat a set of statements a finite number of times.
- Iterate through a string, list and tuple.

- When we use **while** loop to repeat a set of statements till a condition remains True, it means that when we do not know before-hand how many times the statements are to be executed.

```
num = int(input('Enter a number: '))
while num != 5 :
    print(num, num * num)
    num = int(input('Enter a number: '))
```

The loop terminates when 5 is entered as input.

- We can use a **while** loop to repeat a set of statements a finite number of times.

```
count = 0
while count < 10 :
  print(count, count * count, count * count * count)
  count += 1
```

- A **while** loop can also be used to iterate through a string, a list or a tuple using an index value as shown in the following program:

```
s = 'Mumbai'
lst = ['desert', 'dessert', 'to', 'too', 'lose', 'loose']
tpl = (10, 20, 30, -20, -10)
i = 0
while i < len(lst) :
  print(i, s[i], lst[i], tpl[i])
  i += 1
```

Since items in a set or a dictionary cannot be accessed using an index value, it is better to use a **for** loop to access their elements.

- Of the three usages of while loop shown above, the most popular is the first usage—repeat statements an unknown number of times. The other two situations are usually handled using a **for** loop.

Usage of *for* loop

- A **for** loop can be used in following two situations:
  - Repeat a set of statements a finite number of times.
  - Iterate through a string, list, tuple, set or dictionary.

- To repeat a set of statements a finite number of times a built-in function **range( )** is used.
- **range( )** function generates a sequence of integers.

  range(10) - generates numbers from 0 to 9.
  range(10, 20) - generates numbers from 10 to 19.
  range(10, 20, 2) - generates numbers from 10 to 19 in steps of 2.
  range(20, 10, -3) - generates numbers from 20 to 9 in steps of -3.

  Note that **range( )** cannot generate a sequence of **float**s.
- In general,

  range(start, stop, step)

  produces a sequence of integers from start (inclusive) to stop (exclusive) by step.
- The list of numbers generated using **range( )** can be iterated through using a **for** loop.

```
for i in range(1, 10, 2) :
    print(i, i * i, i * i * i)
```

- A **for** loop is very popularly used to iterate through a string, list, tuple, set or dictionary, as shown below.

```
for char in 'Leopard' :
   print(char)
for animal in ['Cat', 'Dog', 'Tiger', 'Lion', 'Leopard'] :
   print(animal)
for flower in ('Rose', 'Lily', 'Jasmine') :
   print(flower)
for num in {10, 20, 30, -10, -25} :
   print(num)
for key in {'A101' : 'Rajesh', 'A111' : 'Sunil', 'A112' : 'Rakesh'} :
   print(key)
```

In the first **for** loop in each iteration of the loop **char** is assigned the next value from the string.

Similarly, in the second, third and fourth **for** loop, in each iteration of the loop **animal**/**flower**/**num** is assigned the next value form the list/tuple/set.

Note that in the last for loop we are printing only the keys in the dictionary. Printing values, or printing both keys and values are covered in Chapter 11.

- If while iterating through a collection using a **for** loop if we wish to also get an index of the item we should use the built-in **enumerate( )** function as shown below:

```
lst = ['desert', 'dessert', 'to', 'too', 'lose', 'loose']
for i, ele in enumerate(lst) :
   print(i, ele)
```

break and *continue*

- **break** and **continue** statements can be used with **while** and **for**.
- **break** statement terminates the loop without executing the **else** block.
- **continue** statement skips the rest of the statements in the block and continues with the next iteration of the loop.

Else Block of a Loop

- **else** block of a **while** loop should be used in situations where you wish to execute some statements if the loop is terminated normally and not if it is terminated abruptly.
- Such a situation arises if we are to determine whether a number is prime or not.

```
num = int(input('Enter an integer: '))
i = 2
while i <= num - 1 :
   if num % i == 0 :
      print(num, 'is not a prime number')
      break
   i += 1
else :
   print(num, 'is a prime number')
```

Note the indentation of **else**. **else** is working for the **while** and not for **if**.

- In the following example **else** block will not go to work as the list contains 3, a non-multiple of 10, on encountering which we terminate the loop.

```
for ele in [10, 20, 30, 3, 40, 50] :
   if ele % 10 != 0 :
      print(ele, 'is a not a multiple of 10')
      break
else :
   print('all numbers in list are multiples of 10')
```

P</> Programs

Problem 6.1

Write a program that receives 3 sets of values of p, n and r and calculates simple interest for each.

Program

```
i = 1
while i <= 3 :
    p = float(input('Enter value of p: '))
    n = int(input('Enter value of n: '))
    r = float(input('Enter value of r: '))
    si = p * n * r / 100
    print('Simple interest = Rs. ' + str (si))
    i = i + 1
```

Output

```
Enter value of p: 1000
Enter value of n: 3
Enter value of r: 15.5
Simple interest = Rs. 465.0
Enter value of p: 2000
Enter value of n: 5
Enter value of r: 16.5
Simple interest = Rs. 1650.0
Enter value of p: 3000
```

```
Enter value of n: 2
Enter value of r: 10.45
Simple interest = Rs. 626.9999999999999
```

Problem 6.2

Write a program that prints numbers from 1 to 10 using an infinite loop. All numbers should get printed in the same line.

Program

```
i = 1
while 1 :
    print(i, end = ' ')
    i += 1
    if i > 10 :
        break
```

Output

1 2 3 4 5 6 7 8 9 10

Tips

- **while 1** creates an infinite loop, as 1 is non-zero, hence true. Replacing 1 with any non-zero number will create an infinite loop.
- Another way of creating an infinite loop is **while True**.
- **end = ' '** in **print( )** prints a space after printing **i** in each iteration. Default value of **end** is newline ('\n').

Problem 6.3

Write a program that prints all unique combinations of 1, 2 and 3.

Program

```
i = 1
while i <= 3 :
    j = 1
    while j <= 3 :
        k = 1
```

```
        while k <= 3 :
            if i == j or j == k or k == i :
                k += 1
                continue
            else :
                print(i, j, k)
            k += 1
        j += 1
    i += 1
```

Output

```
1 2 3
1 3 2
2 1 3
2 3 1
3 1 2
3 2 1
```

Problem 6.4

Write a program that obtains decimal value of a binary numeric string. For example, decimal value of '1111' is 15.

Program

```
b = '1111'
i = 0
while b :
    i = i * 2 + (ord(b[0]) - ord('0'))
    b = b[1:]
print('Decimal value = ' + str(i))
```

Output

```
Decimal value = 15
```

Tips

- **ord(1)** is 49, whereas **ord('0')** is 0.
- **b = b[1:]** strips the first character in **b**.

Problem 6.5

Write a program that generates the following output using a **for** loop:

A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P,Q,R,S,T,U,V,W,X,Y,Z,
z,y,x,w,v,u,t,s,r,q,p,o,n,m,l,k,j,i,h,g,f,e,d,c,b,a,

Program

```
for alpha in range(65, 91) :
    print(chr(alpha), end=',')
print( )
for alpha in range(122, 96, -1) :
    print(chr(alpha), end=',')
```

Tips

- Unicode values of alphabets A-Z are 65-90. Unicode values of alphabets a-z are 97-122.
- Each output of print statement ends with a comma.
- Empty **print( )** statement positions the cursor at the beginning of the next line.

[A] Answer the following questions:

(a) When does the **else** block of a **while** loop go to work?

(b) Can **range( )** function be used to generate numbers from 0.1 to 1.0 in steps of 0.1?

(c) Can a **while** loop be nested within a **for** loop and vice versa?

(d) Can a **while**/**for** loop be used in an **if**/**else** and vice versa?

(e) Can a do-while loop be used to repeat a set of statements?

(f) How will you write an equivalent **for** loop for the following:

```
count = 1
while count <= 10 :
    print(count)
    count = count + 1
```

(g) What will be the output of the following code snippet?

```
for index in range(20, 10, -3) :
   print(index, end = '  ')
```

(h) Why should **break** and **continue** be always used with an **if** embedded in a **while** or **for** loop?

[B] Point out the errors, if any, in the following programs:

(a)
```
j = 1
while j <= 10 :
    print(j)
    j++
```

(b)
```
while true :
    print('Infinite loop')
```

(c)
```
lst = [10, 20, 30,  40, 50]
for count = 1 to 5 :
    print(lst[ i ])
```

(d)
```
i = 15
not while i < 10 :
    print(i)
    i -= 1
```

(e)
```
# Print alphabets from A to Z
for alpha in range(65, 91) :
   print(ord(alpha), end=' ')
```

(f)
```
for i in range(0.1, 1.0, 0.25) :
   print(i)
```

(g)
```
i = 1
while i <= 10 :
   j = 1
   while j <= 5 :
      print(i, j )
      j += 1
      break
   print(i, j)
   i += 1
```

[C] Match the following for the values each **range( )** function will generate.

a. range(5)	1. 1, 2, 3, 4
b. range(1, 10, 3)	2. 0, 1, 2, 3, 4
c. range(10, 1, -2)	3. Nothing
d. range(1, 5)	4. 10, 8, 6, 4, 2
e. range(-2)	5. 1, 4, 7

[D] Attempt the following questions:

(a) Write a program to print first 25 odd numbers using **range( )**.

(b) Rewrite the following program using for loop.

```
lst = ['desert', 'dessert', 'to', 'too', 'lose', 'loose']
s = 'Mumbai'
i = 0
while i < len(lst) :
    if i > 3 :
        break
    else :
        print(i, lst[i], s[i])
        i += 1
```

(c) Write a program to count the number of alphabets and number of digits in the string 'Nagpur-440010'

(d) A five-digit number is entered through the keyboard. Write a program to obtain the reversed number and to determine whether the original and reversed numbers are equal or not.

(e) Write a program to find the factorial value of any number entered through the keyboard.

(f) Write a program to print out all Armstrong numbers between 1 and 500. If sum of cubes of each digit of the number is equal to the number itself, then the number is called an Armstrong number. For example, 153 = (1 * 1 * 1) + (5 * 5 * 5) + (3 * 3 * 3).

(g) Write a program to print all prime numbers from 1 to 300.

(h) Write a program to print the multiplication table of the number entered by the user. The table should get displayed in the following form:

```
29 * 1 = 29
29 * 2 = 58
```

…

(i) When interest compounds **q** times per year at an annual rate of **r** % for **n** years, the principal **p** compounds to an amount **a** as per the following formula:

$$a = p (1 + r / q)^{nq}$$

Write a program to read 10 sets of **p, r, n** & **q** and calculate the corresponding **a**s.

(j) Write a program to generate all Pythagorean Triplets with side length less than or equal to 30.

(k) Population of a town today is 100000. The population has increased steadily at the rate of 10 % per year for last 10 years. Write a program to determine the population at the end of each year in the last decade.

(l) Ramanujan number is the smallest number that can be expressed as sum of two cubes in two different ways. Write a program to print all such numbers up to a reasonable limit.

(m) Write a program to print 24 hours of day with suitable suffixes like AM, PM, Noon and Midnight.