7 Console Input/Output

"Input from keyboard, output to screen..."

Contents

- Console Input/Output means input from keyboard and output to screen.

Console Input

- Console input can be received using the built-in **input( )** function.
- General form of **input( )** function is

  s = input('prompt')

  prompt is a string that is displayed on the screen, soliciting a value. **input( )** returns a string. If 123 is entered as input, '123' is returned.
- **input( )** can be used to receive, 1, or more values.

```
# receive full name
name = input('Enter full name')
# separate first name, middle name and surname
fname, mname, sname = input('Enter full name: ').split( )
```

  split( ) function will split the entered fullname with space as a delimiter. The split values will then be assigned to **fname**, **mname**, **lname**.

- If we are to receive multiple **int** values, we can receive them as strings and then convert them to **int**s.

```
n1, n2, n3 = input('Enter three values: ').split( )
n1, n2, n3 = int(n1), int(n2), int(n3)
print(n1 + 10, n2 + 20, n3 + 30)
```

- The same thing can be done using in a more compact manner using a feature called list comprehension. It applies **int( )** function to every element of the list returned by the **split( )** function.

```
n1, n2, n3 = [int(n) for n in input('Enter three values: ').split( )]
print(n1 + 10, n2 + 20, n3 + 30)
```

  The expression enclosed within [ ] is called list comprehension. It is discussed in detail in Chapter 12.

- **input( )** can be used to receive arbitrary number of values.

```
numbers = [int(x) for x in input('Enter values: ').split( )]
for n in numbers :
    print(n + 10)
```

- **input( )** can be used to receive different types of values at a time.

```
data = input('Enter name, age, salary: ').split( )
name = data[0]
age = int(data[1])
salary = float(data[2])
```

Console Output

- Built-in function **print( )** is used to send output to screen.
- **print( )** function has this form:

```
print(objects, sep = ' ', end = '\n', file = sys.stdout, flush = False)
```

This means that by default objects will be printed on screen (sys.stdout), separated by space (sep = ' ') and last printed object will be followed by a newline (end = '\n'). **flush = False** indicates that output stream will not be flushed.

- Python has a facility to call functions and pass keyword-based values as arguments. So while calling **print( )** we can pass specific values for **sep** and **end**. In this case, default values will not be used; instead the values that we pass will be used.

```
print(a, b, c, sep = ',', end = '!')   # prints ',' after each value, ! at end
print(x, y, sep = '...', end = '#')    # prints '...' after each value, # at end
```

Formatted Printing

- There are 4 ways to control the formatting of output:
  (a) Using formatted string literals - easiest
  (b) Using the **format( )** method - older
  (c) C **printf( )** style - legacy
  (d) Using slicing and concatenation operation - difficult

  Today (a) is most dominantly used method followed by (b).

- Using formatted string literals (often called fstrings):

```
r, l, b = 1.5678, 10.5, 12.66
print(f'radius = {r}')
print(f'length = {l} breadth = {b} radius = {r}')

name = 'Sushant Ajay Raje'
for n in name.split( ) :
    print(f'{n:10}')                    # print in 10 columns
```

- Using **format( )** method of string object:

```
r, l, b = 1.5678, 10.5, 12.66
name, age, salary = 'Rakshita', 30, 53000.55

# print in order by position of variables using empty {}
print('radius = {} length = {} breadth ={}'.format(r, l, b))
print('name = {} age = {} salary = {}'.format(name, age, salary))

# print in any desired order
print('radius = {2} length = {1} breadth ={0}'.format(r, l, b))
print('age={1} salary={2} name={0}'.format(name, age, salary))

# print name in 15 columns, salary in 10 columns
print('name = {0:15} salary = {1:10}'.format(name, salary))

# print radius in 10 columns, with 2 digits after decimal point
print('radius = {0:10.2f}'.format(r))
```

On execution, the above code snippet will produce the following output:

```
radius = 1.5678 length = 10.5 breadth =12.66
name = Rakshita age = 30 salary = 53000.55
radius = 12.66 length = 10.5 breadth =1.5678
age=30 salary=53000.55 name=Rakshita
name = Rakshita       salary =   53000.55
radius =      1.57
```

P</> Programs

Problem 7.1

Write a program to receive radius of a circle, and length and breadth of a rectangle in one call to **input( )**. Calculate and print the circumference of circle and perimeter of rectangle.

Program

```
r, l, b = input('Enter radius, length and breadth: ').split( )
radius = int(r)
length = int(l)
breadth = int(b)
circumference = 2 * 3.14 * radius
perimeter = 2 * ( length + breadth )
print(circumference)
print(perimeter)
```

Output

```
Enter radius, length and breadth: 3 4 5
18.84
18
```

Tips

- **input( )** returns a string, so it is necessary to convert it into int or float suitably, before performing arithmetic operations.

Problem 7.2

Write a program to receive 3 integers using one call to **input( )**. The three integers signify starting value, ending value and step value for a range. The program should use these values to print the number, its square and its cube, all properly right-aligned. Also output the same values left-aligned.

Program

```
start, end, step = input('Enter start, end, step values: ').split( )
# right aligned printing
for n in range(int(start), int(end), int(step)) :
    print(f'{n:>5}{n**2:>7}{n**3:>8}')
print( )

# left aligned printing
for n in range(int(start), int(end), int(step)) :
    print('{0:<5}{1:<7}{2:<8}'.format(n, n ** 2, n ** 3))
```

Output

```
Enter start, end, step values: 1 10 2
   1     1       1
   3     9      27
   5    25     125
   7    49     343
   9    81     729

1   1      1
3   9      27
5   25     125
7   49     343
9   81     729
```

Tips

- **{n:>5}** will print n right-justified within 5 columns. Use < to left-justify.
- {**0:<5}** will left-justify 0^{th} parameter in the list within 5 columns. Use > to right-justify.

Problem 7.3

Write a program to maintain names and cell numbers of 4 persons and then print them systematically in a tabular form.

Program

```
contacts = {
                'Dilip' : 9823077892, 'Shekhar' : 6784512345,
                'Vivek' : 9823011245, 'Riddhi' : 9766556779
            }
for name, cellno in contacts.items( ) :
    print(f'{name:15} : {cellno:10d}')
```

Output

```
Dilip          : 9823077892
Shekhar        : 6784512345
Vivek          : 9823011245
Riddhi         : 9766556779
```

Problem 7.4

Suppose there are 5 variables in a program—**max**, **min**, **mean**, **sd** and **var**, having some suitable values. Write a program to print these variables properly aligned using multiple fstrings, but one call to **print( )**.

Program

```
min, max = 25, 75
mean = 35
sd = 0.56
var  = 0.9
print(
        f'\n{"Max Value:":<15}{max:>10}',
        f'\n{"Min Value:":<15}{min:>10}',
        f'\n{"Mean:":<15}{mean:>10}',
        f'\n{"Std Dev:":<15}{sd:>10}',
        f'\n{"Variance:":<15}{var:>10}' )
```

Output

```
Max Value:          75
Min Value:          25
Mean:               35
```

```
Std Deviation:   0.56
Variance:         0.9
```

Problem 7.5

Write a program that prints square root and cube root of numbers from 1 to 10, up to 3 decimal places. Ensure that the output is displayed in separate lines, with number center-justified and square and cube roots, right-justified.

Program

```
import math
width = 10
precision = 4
for n in range(1, 10) :
    s = math.sqrt(n)
    c = math.pow(n,1/3)
    print(f'{n:^5}{s:{width}.{precision}}{c:{width}.{precision}}')
```

Output

```
 1        1.0       1.0
 2      1.414       1.26
 3      1.732      1.442
 4        2.0      1.587
 5      2.236       1.71
 6      2.449      1.817
 7      2.646      1.913
 8      2.828        2.0
 9        3.0       2.08
```

[A] Attempt the following questions:

(a) How will you make the following code more compact?

```
print('Enter ages of 3 persons')
age1 = input( )
age2 = input( )
```

```
age3 = input( )
```

(b) How will you print "Rendezvous" in a line and retain the cursor in the same line in which the output has been printed?

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

```
l, b = 1.5678, 10.5
print('length = {l} breadth = {b}')
```

(d) In the following statement what do > 5, > 7 and > 8 signify?

```
print(f'{n:>5}{n ** 2:>7}{n ** 3:>8}')
```

(e) What will be the output of the following code segment?

```
name = 'Sanjay'
cellno = 9823017892
print(f'{name:15} : {cellno:10}')
```

(f) How will you print the output of the following code segment using fstring?

```
x, y, z =10, 20, 40
print('{0:<5}{1:<7}{2:<8}'.format(x, y, z))
```

(g) How will you receive arbitrary number of floats from keyboard?

(h) What changes should be made in

```
print(f'{'\nx = ':4}{x:>10}{'\ny = ':4}{y:>10}')
```

to produce the output given below:

```
x =      14.99
y =     114.39
```

(i) How will you receive a boolean value as input?

(j) How will you receive a complex number as input?

(k) How will you display **price** in 10 columns with 4 places beyond decimal points? Assume value of price to be 1.5567894.

(l) Write a program to receive an arbitrary number of floats using one **input( )** statement. Calculate the average of floats received.

(m) Write a program to receive the following using one **input( )** statement.

Name of the person
Years of service
Diwali bonus received

Calculate and print the agreement deduction as per the following formula:

deduction = 2 * years of service + bonus * 5.5 / 100

(n) Which import statement should be added to use the built-in functions **input( )** and **print( )**?

(o) Is the following statement correct?

```
print('Result = ' + 4 > 3)
```

(p) Write a program to print the following values

a = 12.34, b = 234.39, c = 444.34, d = 1.23, e = 34.67

as shown below:

```
a =    12.34
b =   234.39
c =   444.34
d =     1.23
e =    34.67
```

[B] Match the following pairs:

a. Default value of sep in print( )	1. ' '
b. Default value of end in print( )	2. Using fstring
c. Easiest way to print output	3. Right justify num in 5 columns
d. Return type of split( )	4. Left justify num in 5 columns
e. print('{num:>5}')	5. list
f. print('{num:<5}')	6. \n