ProgrammingLesson-19-09

September 21, 2020

1 Welcome to our 2nd class

The Current document gives out a summary of everything covered during this class.

1.1 Summary

1. Questions from previous class
2. Control Flow
    1. Conditionals
    2. Loops
3. Practice problems.

Let's get started

1. Questions from previous class.
   How to take input as decimal number instead of int?

In python `input()` function takes in value from the user as a string. This can be converted to variables of other types by using `type_name()`. Here `type_name` cane be `int`, `float` or `str`.
Examples:

```
int(3.14)
float(3)
str(3.5)
```

You can check the output of the above functions and see what each of them gives as a result. In case of the taking a decimal number as a input, we use `float()` on the `input` function.
This is illustrated in the following code.

```
In [9]: # following code takes in two decimal numbers as input and prints their sum.

        a = float(input("Enter first nunmber\n"))
        b = float(input("Enter second number\n"))
        print(a+b)
```

```
Enter first nunmber
4.5
Enter second number
5.6
10.1
```

2. Control Flow
   Control Flow is the order in the statments in a program is executed. Till now, we have seen that the statements are executed step by step. There are two ways to change this in python. Before understanding these approaches, let's first learn about a boolean expressions.
   Boolean Expression

Boolean expression are those statements in python which when executed give either `True` or `False` as a result. Following code snippets are example of this.

```
In [7]: 6 > 5

Out[7]: True

In [8]: 4 > 5

Out[8]: False

In [9]: 5<=5

Out[9]: True

In [11]: 'Mateo' == 'Mateo'

Out[11]: True
```

As can be seen above, in boolean expression, we are comparing two variables through operators such as >(greater than), <(less than), ==(equal to), `!=`(not equal too), >=(greater than or equal to), <(less than or equal to).

We can also combine mulitiple boolean expressions into a single statement by using `or` and `and` operators. These examples give how they work together.

```
In [15]: print(2 > 1 and 3 > 4)
         print(2 > 1 or 3 > 4)

False
True
```

`and` checks whether both comparisons are true, and outputs True only when both boolean expressions are valid.
`or` outputs True if one of the boolean expression is True, and ouputs False only when both the boolean expressions are False.

Going back to control flow, we use the knowlegde of Boolean expressions. As dicussed earlier there are two approaches to influence control flow, and these are through
1. Conditionals
Conditionals help in making decisions based on the truth of various boolean expressions. This approach is useful when we have multiple statements, but need to executes only one result based what the given input is. The following code illustrates an example.

```
In [16]: Grocery = input("Hey, Mateo! what do you see in the store?\n")
         if Grocery == 'eggs':
             print("Okay, Mateo. Buy 10 eggs!")
         if Grocery == "milk":
             print("Okay, Mateo. Buy 2 packets of milk!")

         # The above program checks what is in the grocery, and print a different result each
```

```
Hey, Mateo! what do you see in the store?
eggs
Okay, Mateo. Buy 10 eggs!
```

Conditionals in python are implemented through `if`, `if-else`, and `if elif else` statements. The following code shows an example.

```
In [17]: a = int(input("Enter a nunmber\n"))
         if a > 6:
             print("%d is greater than 6" %(a))
         elif a == 6:
             print("%d is equal to 6" %(a))
         else:
             print("%d is less than 6" %(a))
```

```
Enter a nunmber
78
78 is greater than 6
```

2. Loops

Another way to affect control flow in computation is through loops. Loops refer to a set of statements that are executed repeatedly. Loops are implemented in python using `for` and `while` statements. An example of each are given below.

```
In [18]: # Normal way to print numbers from 1 to 5
         print(1)
         print(2)
         print(3)
         print(4)
         print(5)
```

```
1
2
3
4
5
```

```
In [19]: # Now using while loop
         a = 1
         while a<6:
             print(a)
             a = a+1
```

```
1
2
3
4
5
the end
```

```
In [20]: # using for loop

         for i in range(1,6):
             print(i)
```

```
1
2
3
4
5
```

Just like conditional, loops are executed as well based on the *truth* of boolean expression.
Now let's look at couple of examples to practice

1.1.1 Practice Problems

1. Ask the user to enter a number and check whether the number is even(multiples of 2) or odd(not multiples of 2).
2. Print numbers from 1 to 1000
3. Take input from the user asking for a name, if the first letter of the name is vowel, print the name 5 times, otherwise print it 7 times.

1.1.2 Other platforms to practice:

Apps
Programminghub, Grasshopper
Websites

Topcoder, codechef, devpost

Other ways to practice

I will send you exercise every two days to practice if you have time.

```
In [ ]:
```