

Variables, Console I/O, Functions & Branching

Ahmet Uysal
auysal16@ku.edu.tr

Date: 5th of February

1 Overview

This handout is prepared for KOLT Python Certificate Program. It contains a brief review of this week's topics and exercise questions.

You can download the starter code from [here](#).

You can find the solutions at [here](#) after all the sections are conducted.

2 Review

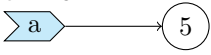
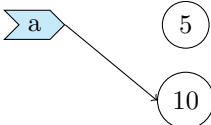
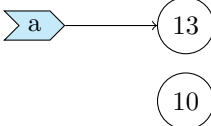
2.1 Variables

Variables are how we represent & store data in Python. You can imagine them as **labels** pointing towards data (Python objects).

2.1.1 Objects

Everything is an object in Python. Even though variables **do not** have **types**, each object has a **fixed type**. Values at the right side of our label analogy are objects!

2.1.2 Example

<pre>a = 5</pre> 	This is an assignment operation. Python first evaluates the right-hand side and creates an integer object to represent this value. After right part is done, Python moves on to left-hand side, creates a label with the name of the variable, and points the label a to this object.
<pre>a = 10</pre> 	This is also an assignment operation. Python creates a new object and points the label a to it. One question is what happens to old integer object? Python automatically deletes object without any references (labels pointing to them).
<pre>a += 3</pre> 	a += 3 is equivalent to a = a + 3 . Therefore, this is also an assignment operation. You know the rest of the story.
<pre>print(a)</pre>	This is a function call . When a function has called, Python evaluates its parameters . The variable a is pointing to integer object 13. print function prints 13 to console.

2.2 Object Types

Type	Explanation	Examples
int	represent integers	3, 4, 17, -10
float	represent real numbers	3.0, 1.11, -109.123123
bool	represent boolean truth values	True, False
str	A sequence of characters.	'Hello', '', '3'
NoneType	special and has one value, None	None

You can learn the `type` of an object by calling the `type` function.

2.3 Console I/O

```
print(*args, sep=' ', end='\n')
```

- Can take arbitrary number of arguments
- Separates elements with single space by default
- Adds newline character `'\n'` to end by default

```
input([prompt])
```

- Prints the prompt to Console
- Program is paused until user enters something
- **returns an str object!**

2.4 Simple Functions

Functions are blocks of **organized**, **reusable** code that carry some **specific** tasks. We will talk more about functions in more detail soon, however, we wanted to briefly introduce the most simple form of functions to help you write better Python code. In Python, indentation is used to group different expressions. Recommended amount for an indentation level is 4 spaces.

```
1 def function_name():
2     <expression>
3     <expression>
4     <expression>
5     ...
```

Code snippet 1: Syntax for defining a simple function

2.5 Branching

We might want to control how our program behaves based on different conditions. A *condition* can be any Python expression, Python will evaluate this expression and decide on which `code block` will be executed based on whether this expression has a **falsy** or **truthy** value.

```

1 if <condition>:
2     <expression>
3     <expression>
4     ...

```

Code snippet 2: if statement, code block will be executed only if the condition is **truthy**.

```

1 if <condition>:
2     <expression>
3     <expression>
4     ...
5 else:
6     <expression>
7     <expression>
8     ...

```

Code snippet 3: if-else statement. If condition is a **truthy** value, first code block (if block) is executed. Otherwise, the second block (else block) is executed.

```

1 if <condition>:
2     <expression>
3     <expression>
4     ...
5 elif <condition>:
6     <expression>
7     <expression>
8     ...
9 ...
10 else:
11     <expression>
12     <expression>
13     ...

```

Code snippet 4: if-elif-else statement. You can add as many **elif** blocks as you want. Think about the difference between having **elif** blocks and **if** blocks.

```

1 # 'Falsy' values
2 bool(None) # => False
3 bool(False) # => False
4 bool(0) # => False
5 bool(0.0) # => False
6 bool('') # => False
7 # Empty data structures
8 bool([]) # => False
9 #####
10 # Everything else is 'truthy'
11 bool(-100000) # => True
12 bool('False') # => True
13 bool(3.14) # => True
14 bool(int) # => True
15 # Nonempty data structures
16 bool([1, 'a', []]) # => True
17 bool([False]) # => True

```

Code snippet 5: Truthy and Falsy values.

3 Exercises

3.1 FizzBuzz

FizzBuzz is a group game that is usually played to teach division to children. In computer science, it is also a popular interview question. We will implement a slightly modified version of it.

Your program should take an integer from the user. Then according to the input, it should print:

- an error message 'You entered a negative number!' if the number is negative,
- Fizz, if the number is a multiple of 3
- Buzz, if the number is a multiple of 5
- FizzBuzz, if the number is a multiple of 15
- The number itself otherwise

3.2 Leap Year

Roman general Julius Caesar introduced the first leap years over 2000 years ago. But the Julian calendar had only one rule: any year evenly divisible by four would be a leap year. This formula produced way too many leap years. Still, it was not corrected until the introduction of the Gregorian calendar more than 1500 years later.

In the Gregorian calendar, three criteria must be taken into account to identify leap years:

- The year can be evenly divided by 4;
- If the year can be evenly divided by 100, it is NOT a leap year, unless;
- The year is also evenly divisible by 400. Then it is a leap year.

Write a program that takes a year from the user and prints whether it is a leap year or not in the Gregorian calendar.

3.3 Birthday

Have you ever wondered how many days you have to wait for your birthday celebrations after you celebrate New Year? Our team member Necla wants to learn the answer of this question, but she is too busy preparing for the upcoming lecture. Help Necla write a program for this, so that she can spend her time to plan an awesome lecture instead.

Write a program that first takes the year, month (as integer), and day of the user's birth date one by one. Then, calculate and print how many days your birthday is away from the beginning of the year.

References

- [1] Fizz buzz. (2020). Retrieved 4 February 2020, from https://en.wikipedia.org/wiki/Fizz_buzz
- [2] Leap Day on February 29. (2020). Retrieved 4 February 2020, from <https://www.timeanddate.com/date/leapyear.html>