

# Skill-02-IntroPython-Part2-Student

September 12, 2024

## 1 Introduction to Python, Part 2

This skill homework shows a few extensions to the previous homework on variables, lists, and functions, then covers some completely new territory.

### 1.1 More on Operators

Python, of course, has the traditional mathematical operators  $+$ ,  $-$ ,  $*$ , and  $/$  for addition, subtraction, multiplication and division. The one important note is that division with integers (since we are using python 3) will always give a float.

In the cell below try some integer division like

```
print(33/33)
print(33/2)
print(33//2)
```

### 1.2 Additional Mathematical Operators

There are three additional math operators you should know.

The remainder operator,  $\%$ , does a division and returns the remainder. It works with both integer and float division. Try the following remainders:

```
print(255 % 16)
print(4.2%1.5)
```

Next the power operator is  $**$ . (*Many languages use the  $\wedge$ , but in python this is the bitwise XOR operator, so be careful!*) It raises the first number to the power of the second. If both operators are integers, it will return an integer if possible. Try the following:

```
print(2**3)
print(3**2)
print(2**(-1))
print(1.2**1.2)
```

**Note:** I usually do not put spaces around the power operator because it has higher precedence than the other mathematical operators.

Finally, the *integer division* or *floor division* operator is  $//$ . This acts like normal integer division if the operators are integers. The odd feature is that the answer is rounded *down*, even if the answer is negative. If either or both of the operands are floats, the answer will be a float round down. Try the following floor divisions:

```
print(3//2)
print(-3//2)
print(100.0//3.0)
```

Look at the results to make sure you understand how these operators work.

[ ]:

### 1.3 Compound Assignments Like +=

Like C and most modern languages, python supports *compound assignments*. These are statements that modify the current value of a variable by performing an operation on it. They are equivalent to assigning the result of an operation to the first operand:

The following pairs of statements have the same result:

`y += x` or `y = y + x`

even

`price *= units + 1` is the same as `price = price * (units+1)`

In the code cell below, create some variables and try compound assignments on them. Here are my suggestions, but feel free to try your own.

```
x = 2
y = 3
y += x
x *= 1.6
y += x
print(x)
print(y)
```

### 1.4 Are You Remembering Tab Completion?

Keep trying tab completion. Once you get used to it, you will regret using a software development system that doesn't have it.

[ ]:

## 2 Boolean Variables and Operators (True & False)

You often need to make true or false decisions in computer code. This is often done with *Boolean* or true-false expressions. In python the two values are **True** and **False**. Boolean variable can also be defined. In addition zero is **False** and a non-zero number is **True**

Some Boolean operators are **not**, **and**, and **or**. Note that in python, these are spelled out.

In the cell below create the variable **tr** with the value **True**, and the variable **fa** with the value **False**. Then print out **not tr**, **tr and fa**, and **tr or fa**. Make sure you understand the results before moving on.

My examples are

```
tr = True
fa = False
print(not tr)
print(not fa)
print(tr and fa)
print(tr or fa)
```

[ ]:

### 3 Comparison Operators ==, >, etc.

Python has the usual comparison operators == (for equal to), != (not equal to), and the math-type comparisons >=, >, <, <=. All of these operators evaluate to either `True` or `False`. In the cell below try each of these by typing something like

```
print(3 < 2)
```

Make one example of each comparison operator.

[ ]:

### 4 Flow control with if, elif, and else

The following is an example of an if statement:

```
answer = 3
if answer > 2:
    print("Greater than 2")
    print("Go fish")
print("Done")
```

Note the `:` after the `if` statement. This is required. Also note that the lines indented are only executed if the expression is true and the line dedented to the same position as the `if` statement is always executed. This is a reminder that indentation is very important in python.

Go ahead and run the lines above in the code cell below.

[ ]:

#### 4.1 The else Statement

Many times you want to choose between two strategies. The `else` statement works like this:

```
answer = 5
if answer > 2:
    print("Greater than the mystery number")
    print("Go fish")
else:
    print("Less than or equal to the mystery number")
    print("Try another guess")
```

```
print("Done")
```

Again, the block that is executed when the `if` expression is false must be indented. Try this in the cell below. Rerun the cell with different values for `answer`.

[ ]:

## 5 Multiple Choices. The `elif` Statement

Finally if there are multiple branches you can use the `elif` statement which stands for *else if condition*. You can stack together multiple `elif` statements in after an `if` statement. Here is an example:

```
answer = 2
if answer > 2:
    print("Guess is too big")
    print("Go fish")
elif answer == 2:
    print("Correct! Game Over!")
else:
    print("Guess is too small")
    print("Try another guess")
print("Done")
```

Enter this code and play the game using different guesses for `answer`.

[ ]:

## 6 Make This a Function

Many times I will develop code in a cell (usually more complicated than our guessing game), then when it is working make it into a function. In the cell below, write a function with one argument being the guess, then test your code by guessing answers that are too small, too big and just right. Define the answer as a global variable in the cell. You also want your function to return `True` when the guess is correct and `False` when the answer is wrong.

I have given you an outline for the function. You have to fill it in and test it.

BTW, the `#` symbol starts a comment. The `#` and anything after it in a line is ignored.

**Note: The code in the cell below WILL NOT WORK. You have to fix it to make it run.**

[ ]:

```
# Make a function called game
answer = 4

# next start the function definition
# print the value guessed
print("Your guess is ", guess)
if guess > answer:
```

```

    print("Your guess is too big")
    return False
elif guess < answer:
    print("Your guess is too small")
    return False
else:
    print("Correct! The answer is", answer)
    return True

print(game(1))
print(game(10))
print(game(4))

```

## 7 while Loops

Many times your code has to loop until some ending condition is met. Now that you have your function `game`, here is how you can get input from the user, play the game, and quit when the answer is right. I also generate a random integer from 1 to 10.

The code in the cell is not quite right. You should change it to correctly keep track of the number of guesses.

**Note:** this code uses the function `game` you fixed above.

```

[ ]: from random import randint
    answer = randint(1,10)
    print("Guess a number from 1 to 10")
    yourGuess = int(input("Enter your guess: "))
    while game(yourGuess) == False:
        yourGuess = int(input("Enter your guess: "))
    print("It took you", nGuesses, "guesses to get it right!")

```

## 8 Other Topics

There are many other topics, but the material covered should get you started.

## 9 Conclusion

This ends the second skills homework for python. This tour through operators and function should get you started. Feel free to contact me with any questions by e-mail.

The next topics are libraries, arrays from the `numpy` library, and reading and writing data files.