Installation, Basic Syntax, Variable Types, Basic Operators

Ramkumar N

Contents

Before we begin, I hope all of you have a computer and have setup Python3 development environment on your computer. If not follow this video and please get it setup (https://www.youtube.com/watch?v=UvcQlPZ8ecA). If you do not have python setup on your computer you can download a python3 interpreter for your android phone/tablet by clicking this link
https://play.google.com/store/apps/details?id=ru.iiec.pydroid3&hl=en_IN&gl=US. But do note it will be very difficult for you to program on you phone. Please get buy/borrow laptop ASAP!

1 Why learn Python? What are the characteristics of Python?

1. **Beginners Language:** Python is a beginners language. The syntax is fairly intuitive and easy to learn. The language has so many libraries for everything including 1) mathematical computations, plotting, visualizations, 2) gaming, 3) web development, 4) IoT etc.
2. **Interpreted Language:** Program does not need to be compiled or error checked before running. The code will run up to the point it encounters an error. I will demo this a bit later in class today.
3. **Object Oriented Language:** Supports the full gammut of object oriented concepts. We will learn more about this later.

Last week we also ran the 'Hello World' program.

```
print("Hello, World!")
```

While doing that, you might have opened this window (1).

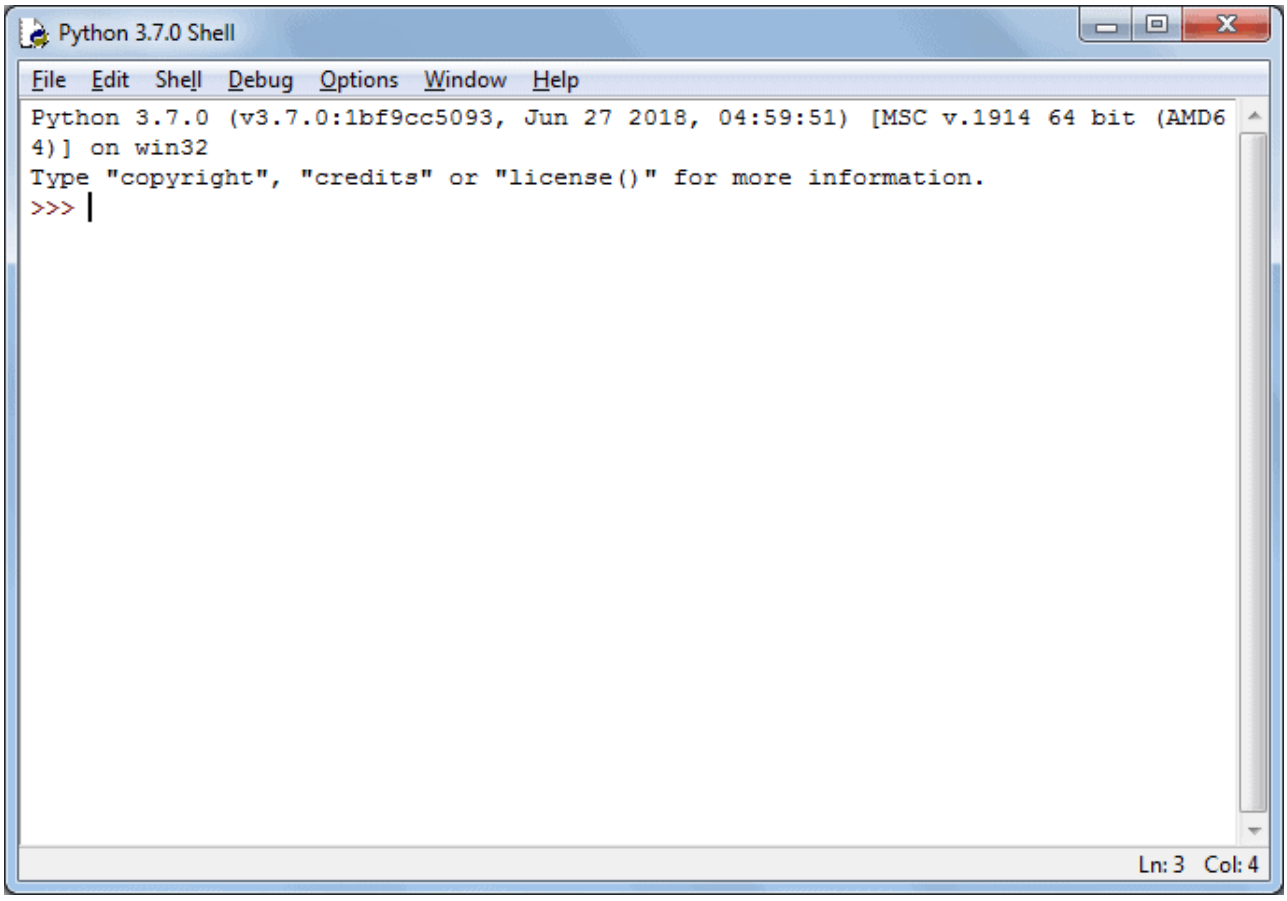

Figure 1: Python IDLE shell

This is called the 'Python Shell'. A shell is a interactive program used for gathering user inputs and outputting results onto the screen. Similar to python shells, there are several shell types - powershell in windows, perl, matlab, bash, c shell etc., are some other varieties. On this shell, we can write a command and it outputs the results - just like the 'print("Hello, world")' command while executing the helloworld program. A python shell is indicated by the >>> prompt. A shell also remembers the previous state of the code when executing the next. For instance if you type the following code in the shell

```
>>> a = 5
>>> print(a)
```

But when writing long and complicated programs, we do not use the shell, we write what is called a 'script'. A script is a sequence of commands that are executed one after the other by the python shell. To create a script, click File → New File on the IDLE window. Note, the extension of the file is '.py'. This helps the operating system understand what program the file is supposed to be associated with. In our case 'python'. If it had a '.mp3' extension, then the associated program would have been perhaps vlc.

2 Variables

Variables are also called identifiers. There are some rules for defining a variable

1. A variable should start with upper/lower case (A-to-Z or a-to-z) or an (_) underscore followed by one or more letters or numbers.
2. Characters such as '.', '$' or any such special characters are not allowed inside variables.
3. Do not use any reserved keywords as a variable name. A list of keywords are in the table below (2).

and	exec	not
as	finally	or
assert	for	pass
break	from	print
class	global	raise
continue	if	return
def	import	try
del	in	while
elif	is	with
else	lambda	yield
except		

Figure 2: list of keywords - source: https://www.tutorialspoint.com/python3/python_basic_syntax.htm

Much later you will learn about object oriented programming (classes, objects and such). We will refine the rules for variable definition a bit more then.

2.1 How do I work with variables?

Variables use what is called the '=' equal to or assignment operator. In maths, '=' equal to signifies equality - what is on the left hand side is equal to what is on the right. However, in computer languages, '=' has a different connotation. It means assigning the value that on on the right to the variable on the left. Variables can be assumed to be like containers. You take what is on the right and put it in the container/variable on the left.

```
myvariable = 5
```

We are putting the value 5 into the variable named - 'myvariable' by using the assignment operator. Now you can use the variable name handle 'myvariable' anywhere you want.

NOTE: You can assign same values for multiple variables like this.

```
a=b=c=10
```

3 Lines and Indentation

3.1 Indents

Remember in the last class we discussed something about indentation? Indent simply means a block of white spaces. For instance, take the following code

```
a = 5
i = 0
while(i<a):
   print("hello world")
   i=i+1 # this means increment the value of i by 1; mathematically this makes
no sense
```

Do not worry about the syntax of the code..... Please ignore the content and just focus on the indent!. Notice the 'print("hello world")' and 'i=i+1' are indented. Here I have used a tab space. Python recommends using 4 white spaces for indents but for ease of use, you can use tabs. Indents signify that the two lines form a block of code nested under the 'while(i<a):' statement.

3.2 Comments

Notice the blue coloured line in the above block of code preceded by a '#' - hash symbol. This signifies comments. Comments are simply helper lines that you put to describe something for yourself (author of the code or others). Comments are ignored by the python interpreter. ***It is super important to add a lot of comments to your code. Else, later when you revisit your code you won't be able to make head or tail out of it.***. Since the python interpreter does not read commented parts of the code, we can use comments to our advantage while writing. Many a times when writing lines of code or debugging, we would like to block out certain lines of code from

temporary execution. We can comment out those lines too - this is somewhat a popular hack to get the compiler to ignore a specific set of lines.

3.3 Multi-line Statements

Suppose you are writing code that spans over multiple lines. Its such a pain to write it in a single line as the code becomes unreadable. We can wrap the text into multiple lines using the \character. For example

```
total = Item_1 + \
      Item_2 + \
      Item_3 # Though they span 3 lines, pythonically they are a single line.
```

3.4 Quotations

There are a specific class of entities in python (or any programming language for that matter) called strings. Strings can be enclosed in single, double or triple quotes as in the example below 3.

```
word = 'word'
sentence = "This is a sentence."
paragraph = """This is a paragraph. It is
made up of multiple lines and sentences."""
```

Figure 3: strings; source: https://www.tutorialspoint.com/python3/python_basic_syntax.htm

3.5 Blank Lines

Full blank lines are ignored by the python interpreter.

3.6 Multiple statements

As you noticed in all the cases above, we insert one statement in every pythonic line. We can put many lines in one statement also. But they have to be separated by ';'.

```
a = 5; b = 10; print(a); print(b)
```

But as you notice, this is hard to read hence not widely used.

4 Data Types

A data type is as the word describes - the type of the data stored in a variable. There are a few basic data types python supports by default.

1. int
2. float
3. string
4. complex numbers

The above three are the most commonly used data types in python. An 'int' signifies data type that stores any integer. 'float' stores a floating point or numbers with decimal points. 'strings' stores character/words/phrases/sentences.

```
a = 10 # assignment of int
b = 3.0 # assignment of float
c = "hello" # assignment of string
d = 3+4j # assignment of complex numbers
```

We can check the datatype stored in a variable if need be using the type() function

```
a = 10
print(type(a))
```

The first two form the number data types (obviously duh!). You are allowed to do certain operations on those datatypes. Those operation include

1. arithmetic operations
2. comparison operations
3. logical operations
4. membership operations
5. bitwise operations

These are the most famous ones.

4.1 arithmetic operations

These are the common addition, subtraction, multiplication, division, modulo (this gets you the remainder), exponent and floor division operations. Lets see them in action.

```
# addition
a = 5 + 6
b = a + 3
c = a + b
print(a)
print(b)
print(c)

# subtraction
a = 5 - 3
b = a - 10
print(a)
print(b)

# Multiplication
a = 5 * 3 # notice * star operator signifies multiplication

# Division
a = 5/3
print(a) # see what the output is

# modulo
a = 5%3
print(a) # see what the output is

# exponent
a = 5**2
b = a**3

# floor division
a = 5//3
print(a) # see what the output is
```

4.2 Comparison Operations

Lets try some comparison operations. Some comparison operations include

1. equality check
2. greater than check
3. lesser than check
4. greater than or equal to check
5. lesser than or equal to check
6. not equal to check

Lets see how they work in code

```
a = 3
b = 4

print(a==b) # check if equal to
print(a>b) # check for greater than
print(a<b) # check for lesser than
print(a<=b) # check if equal to or greater than
print(a>=b) # check if lesser than or equal to
print(a!=b) # check if not equal to
```

4.3 Logical Operations

Logical operators check for pairwise logical equivalence. Usually this is done for boolean variables. Meaning data can take the value 1 or 0 (or 'True' or 'False').

1. AND operator
2. OR operator
3. NOT operator

```
print(1 and 0)
print(1 and 1)
print(1 or 0)
print(not(1)) # this operator inverts the value

# note instead of 1 or 0, we can use True or False
print(True and True)
print(not(True))
```

4.4 Membership Operations

You have not learnt lists or tuples yet. I will give a small example though. A list is nothing but a collection of variables as in the example below. There are only two relational operators.

1. in - is present
2. not in - is not present

```
a = [1, 2, 3, 4, 5] # a list of integers
b = 3

print(b in a) # True
print(b not in a) # False
```

4.5 Bitwise Operations

They perform bitwise operations on the variables. The operators include

1. & - and operator
2. | - or operator
3. $\wedge$ - xor operator
4. $\sim$ - not operator

Just like above, you can try these out.

4.6 Deleting an assigned variable

Here is how you delete an assigned variable. Using the 'del' operator.

```
a = 10
print(a)
del a # del operator deletes a defined variable
print(a)
```

Note: All rules for the operators are not depicted here. You will practice a lot solving lab assignments which will help you learn the most popular ones.

5 Some other minute topics

5.1 type casting

As the topic heading suggests you can cast/throw away the type of a variable into another - if it makes logical sense. For example, we can convert an integer into a string as shown in the example below.

```
a = 1 # here a is an integer
print(type(a))
a = str(a) # here we cast the type of a from int to string
print(type(a))
a = int(a) # a which was a string is now converted back to integer
print(type(a))
```

Obviously any string cannot be converted to int/float.

```
a = "hello"
a = int(a) # this will throw up an error
```

Likewise type casting for float can also be done.

5.2 Input handling

Until now we have been dealing with us directly giving variable values in the code. What if we want the user to input the value at runtime? How do we achieve that? This can be done using the input function.

```
a = input("Enter a's value here: ")
print(type(a)) # notice all the inputs are strings. You can type cast them into
whatever datatype you want.
```

6 Practice Problems

1. Write a program to assign the value 5 to a variable and print it. Also print the type of the variable

```
x = 5
print(x)
```

2. Write a program to assign a string to a variable. Also print the type of the variable.

```
x = "hello"
print(type(x))
```

3. Make a simple calculator addition, subtraction, multiplication, division (both quotient and remainder), also rounding off, floor division.

```
x = 10
y = 20

# addition
z = x+y
print("Sum = ", z)

# subtraction
z = x-y
print("Difference = ", z)

# product
z = x*y
print("product = ", z)

# division value
z = x/y
print("division value = ", z)

# round off
remainder = x%10
quotient = x//10
if(remainder>=5):
   quotient = quotient+1
   rounded_off_value = quotient*10
   print("rounded off value is: ", rounded_off_value)
else:
   rounded_off_value = quotient*10
```

```
    print("rounded off value is: ", rounded_off_value)

# floor division
z = x//y
print("floor division value = ", z)
```

4. Add two strings. See if subtraction operation works on strings.

```
a = "hello"
b = " world"
print(a+b)
print("subraction is not defined for strings")
```

5. Type cast two numbers into strings and add them.

```
a = 10
b = 20
a = str(a)
b = str(b)
summ = a+b
print(summ)
```

6. Find out the grade of a student. The grade rule is

   (a) 0-39 - F
   (b) 40-49 - D
   (c) 50-69 - C
   (d) 70-89 - B
   (e) 90-100 - A

   User input the marks and present his/her grade.

```
marks = 59
if(marks < 39):
    print("Sorry, you have failed")
elif(marks < 49):
    print("your grade is D")
if(marks < 69):
    print("your grade is C")
if(marks < 89):
    print("your grade is B")
else:
    print("your grade is A")
```

7. User input a word (password) and check if the user has entered the correct password.

```
mypassword = "hakunamatata"
userpass = input("Enter your password: ")

if(mypassword == userpass):
   print("You have entered the correct password")
else:
   print("The password you have entered is wrong")
```

8. Check if the word is more than ten characters long. Hint: google up 'len()' in python.

```
myword = "hello world"
if(len(myword)>10):
   print("yes my word is more than 10 characters long")
else:
   print("no my word is less than 10 characters long")
```

7 Additional Reading Material

https://amritauniv.sharepoint.com/sites/saraths/SitePages/Problem-Solving.aspx. Read week 1 and 2 videos, slides and additional materials.