BRAC UNIVERSITY

Inspiring Excellence

Course Title: Programming Language II

Course Code: CSE 111

Semester: Summer 2020

Concise Note

Topic: Introduction to variable, IO and Type Conversion

Variables

Variable is a location which is used to store information. Single information can be stored in a variable by assigning a name to the variable.

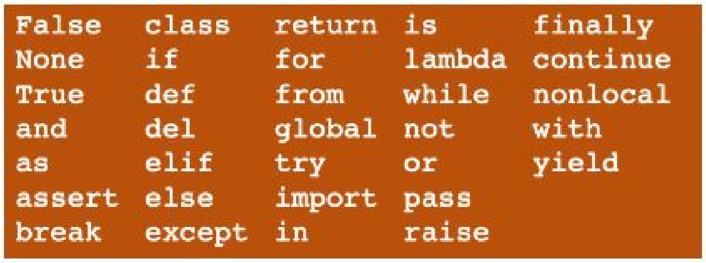
A variable can be of several data types. They are as follows:

- Int - Any whole number or natural number which can be positive or negative.
- Float - Any decimal number.
- Character - Any single digit character which can be alphabet, digit or any symbol.
- String - A line consisting of one or more than one character.
- Boolean - A variable containing only True or False.

A variable can have a name. But the name follows some rules. They are:

1. It cannot start with a numeric digit (0-9).
2. It cannot contain any special character except _ and $.
3. It can start with any alphabet (a-z, A-Z).
4. Rest of the name can have any valid character mentioned above.
5. It's a good practice to start the name with a lowercase letter.
6. Variable names are case sensitive. For example, myname and MyName are not the same variable name.
7. For more than one word in a variable, use an underscore between them. For example, my_name, my_bracu_id etc.

No default keywords of Python can be used as a variable name. A list of python keywords are



False	class	return	is	finally
None	if	for	lambda	continue
True	def	from	while	nonlocal
and	del	global	not	with
as	elif	try	or	yield
assert	else	import	pass	
break	except	in	raise	

Variable declaration

Unlike other programming languages like Java, C++, Python variables don't require any data type while declaration. Every variable in Python is an object. For example

```
>> my_name  = ‘xyz’
>> my_id = 20000000
```

To delete a variable, one can write del before a variable. For example

```
>> del my_name
```

Input from the user

Taking input in python is very simple. Syntax to take an input is given below:

```
>> name = input(‘Write your input prompt here’)
```

This input function returns a String. Therefore, any given input gets saved to the variable as a String. To use the input as Data Types, it needs to be casted. For example,

```
>> num = input(‘Give a number’)
>> num = int(num)
```

A variable in python can hold any data type and thus a string carrying variable can also hold an integer or any other data types. It will carry the latest value assigned.

Type conversion in Python

Some type conversion or casting in Python are shown below:

- ➔ *int(val)* = Converts val variable to integer.
- ➔ *int(val,base)* = Converts val to the given base like Binary, Octal, Hexadecimal. All numbers are Decimal by default.
- ➔ *str(val)* = Converts val to string
- ➔ *float(val)* = Converts val to float

Output in Python

Any data can be displayed to the console using the *print()* function. For example
>> print('CSE111')
Output: CSE111

To print several variables or values, pass them as comma separated parameters to the *print()* function. *print()* function automatically gives spaces in between the comma separated values.

>> print('CSE110', 'CSE111')
Output: CSE110 CSE111

By default, *print()* separates each object by a single space and appends a newline to the end of the output

General syntax of print function

print() is a function/method that takes some parameters. Python *print()* looks like

```
print(object(s), sep=separator, end=end, file=file, flush=flush)
```

Meaning of the arguments

***object(s)* -** Any object that needs to be printed. This argument can receive multiple objects and converts them to String before printing. For example,

>>print('CSE110', 'CSE111'). Here 'CSE110' and 'CSE111' are two String objects.

***sep* -** Separator separates each object by a given String. By default it separates with spaces ‘ ‘. To specify any other separator, it needs to be mentioned in the argument **sep =** ‘required separator’. For example,

```
>> a =10, b = 20, c = 30
>> print(a,b,sep=’ # ‘)
>> print(c)
Output>> 10 # 20
Output>> 30
```

***end* -** End argument is added to the end of the print statement. By default, new line or ‘\n’ is the default value of the end argument and thus after each print statement, it goes to a new line.To specify any other end argument, it needs to be mentioned in the argument **end=** ‘required end string’. For example,

```
>> a =10, b = 20, c = 30
>>print(a, end =’->’)
>>print(b, end = ‘---’)
>>print(c, end = ‘ endOfprint ’)
Output>> 10->20---30 endOfprint
```

These arguments are optional. If not specified, its default value will be used.
