Lab1-2

Basic syntax, variables, and operators

1. **Mini assignment**
    a. Use triple quotes (""") to denote string literal more than one line with the following details: firstname, lastname, and address
    <<fill your result>>

2. **Mini assignment**
    a. Use multi-line statement to print your firstname and lastname
    <<fill your result>>

3. **Mini assignment**
    a. displays the prompt, the statement saying “hello <<your ID>> <<your full name>>
    <<fill your result>>

4. See how multiple commands can be written in one line
    import sys; x = 'foo'; sys.stdout.write(x + '\n')
    <<fill your result>>

5. **Mini assignment**
    a. Use help function for "print" command
    help(print)

    b. Use option "sep" and "end" to see the result
        i. Use "," for "sep" and use "-" for "end"
        <<fill your result>>

6. **Mini assignment**
    a. Use help function for "sys.stdout.write" command
    (This is not straightforward, try yourself Hint: use quote)
    <<fill your result>>

7. **Mini assignment**
    a. Print the following values one by one

```
counter = 100          # An integer assignment
miles   = 1000.0       # A floating point
name    = "John"       # A string
```

<<fill your result>>

b. Print the following values one by one

```
a,b,c = 1,2,"john"
```

<<fill your result>>

c. Print the following values one by one

```
a = b = c = 1
```

<<fill your result>>

8. **Mini assignment (String)**

   a. Fill the following output (Note: use parenthesis after print command)

```
str = 'Hello World!'

print str          # Prints complete string
print str[0]       # Prints first character of the string
print str[2:5]     # Prints characters starting from 3rd to 5th
print str[2:]      # Prints string starting from 3rd character
print str * 2      # Prints string two times
print str + "TEST" # Prints concatenated string
```

<<fill your result>>

9. **Mini assignment (List)**

   a. Fill the following output

```
list = [ 'abcd', 786 , 2.23, 'john', 70.2 ]
tinylist = [123, 'john']

print list          # Prints complete list
print list[0]       # Prints first element of the list
print list[1:3]     # Prints elements starting from 2nd till 3rd
print list[2:]      # Prints elements starting from 3rd element
print tinylist * 2  # Prints list two times
print list + tinylist # Prints concatenated lists
```

<<fill your result>>

Try

l1 = [1,2,3]

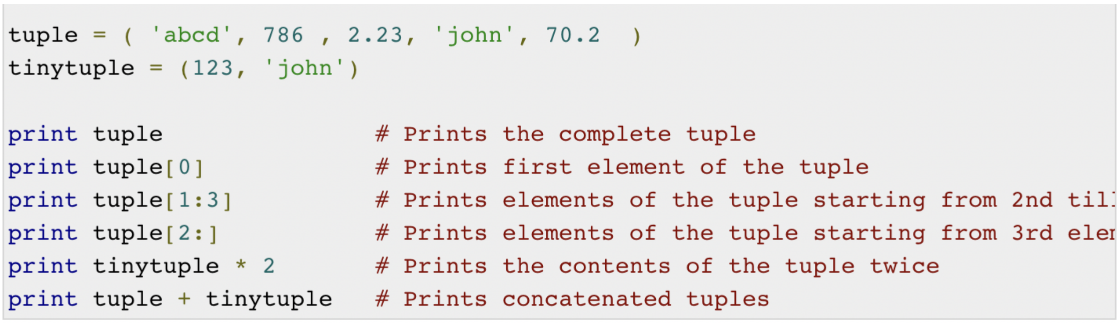
l2 = ['a','b','c']

l1 + l2

<<fill your result>>

*Note + used in list means concatenation

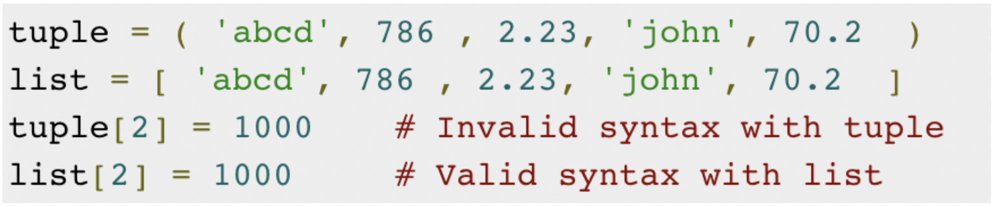
10. **Mini assignment (Tuple)**
    a. Fill the following output



```
tuple = ( 'abcd', 786 , 2.23, 'john', 70.2  )
tinytuple = (123, 'john')

print tuple              # Prints the complete tuple
print tuple[0]           # Prints first element of the tuple
print tuple[1:3]         # Prints elements of the tuple starting from 2nd til
print tuple[2:]          # Prints elements of the tuple starting from 3rd ele
print tinytuple * 2      # Prints the contents of the tuple twice
print tuple + tinytuple  # Prints concatenated tuples
```

<<fill your result>>

11. **Mini assignment (Tuple vs List)**
    a. Fill the following output and error



```
tuple = ( 'abcd', 786 , 2.23, 'john', 70.2  )
list = [ 'abcd', 786 , 2.23, 'john', 70.2  ]
tuple[2] = 1000    # Invalid syntax with tuple
list[2] = 1000     # Valid syntax with list
```

<<fill your result>>

12. **Mini assignment (Dictionary)**
    a. Fill the following output

```
dict = {}
dict['one'] = "This is one"
dict[2]     = "This is two"

tinydict = {'name': 'john','code':6734, 'dept': 'sales'}


print dict['one']        # Prints value for 'one' key
print dict[2]            # Prints value for 2 key
print tinydict           # Prints complete dictionary
print tinydict.keys()    # Prints all the keys
print tinydict.values() # Prints all the values
```

<<fill your result>>

***Tip: Use "type(x)" to see data type

13. Conversion

src = "12"

a. To integer
re = int(src)
<<fill your result by showing type of re>>

b. To float
re = float(src)
<<fill statement>>

c. To string
src = 1234
<<fill your statement>>

d. Integer to Char
chr(65)
chr(66)
chr(97)
<<fill result>

e. Char to Integer

ord("A")
ord("B")
ord("a")
<<fill result>

f. To tuple
src = "11,22,33,44"
<<fill your result>>

Note: No direct way to convert string into tuple with comma separator.

src = [11, "aa" ," bb" ,"cc"]
<<fill your result>>

b = tuple([1, 2, 3, 4, 5]) # list to tuple
<<fill your result>>

g. To list
src = "11,22,33,44"
li = src.split(",") # , is delimiter
<<fill your result>>

a = list((1, 2, 3, 4, 5)) # tuple to list
<<fill your result>>

h. To set
c = set([1, 2, 3, 4, 5]) # list to set
<<fill your result>>

i. To dictionary
l1 = [1,2,3,4]
l2 = ['a','b','c','d']
d1 = zip(l1,l2)
print(d1)
print(dict(d1))
<<fill your result>>

Note: Try zip() here https://www.w3schools.com/python/ref_func_zip.asp

```
# mapping to dict
d = dict(one = 1, two = 2, three = 3)
```
<<fill your result>>

```
# iterable list to dict
e = dict([('one', 1), ('two', 2), ('three', 3)])
```
<<fill your result>>

14. **Mini assignment (Dictionary)**
    a. Use another way to convert string to dictionary (ask Google)
    src = "{'muffin' : 'lolz', 'foo' : 'kitty'}"
    <<fill your result>>

15. Arithmetic Operators
    Note: for later version of Python, you must use parentheses '()' for print function

```
a = 21
b = 10
c = 0

c = a + b
print "Line 1 - Value of c is ", c

c = a - b
print "Line 2 - Value of c is ", c

c = a * b
print "Line 3 - Value of c is ", c

c = a / b
print "Line 4 - Value of c is ", c

c = a % b
print "Line 5 - Value of c is ", c
```
<<fill your result with explanation of each operator>>

*Note "+" of two strings is concatenation
Try
```
src1 = "a"
src2 = "b"
src1 + src2
```

<<fill your result>>

16. Arithmetic Operators

   a. Provide your example of using "Exponential" and "Floor division"
      <<fill your result >>

   b. What is the output datatype when you use floor division between 2 integers
      <<fill your result >>

17. Arithmetic Operators (learn the meaning of the following statements)

```
a, b = 101, 10
a //= b
```
<<fill your result of a>>

```
a += b
```
<<fill your result of a>>

```
a -= b
```
<<fill your result of a>>

```
a *= b
```
<<fill your result of a>>

```
a /= b
```
<<fill your result of a>>

18. Comparison Operators

```
a = 21
b = 10
```
try
```
a == b
```
<<fill your result>>

```
a != b
```
<<fill your result>>

```
a < b
```
<<fill your result>>

a > b

<<fill your result>>

19. Assignment Operators

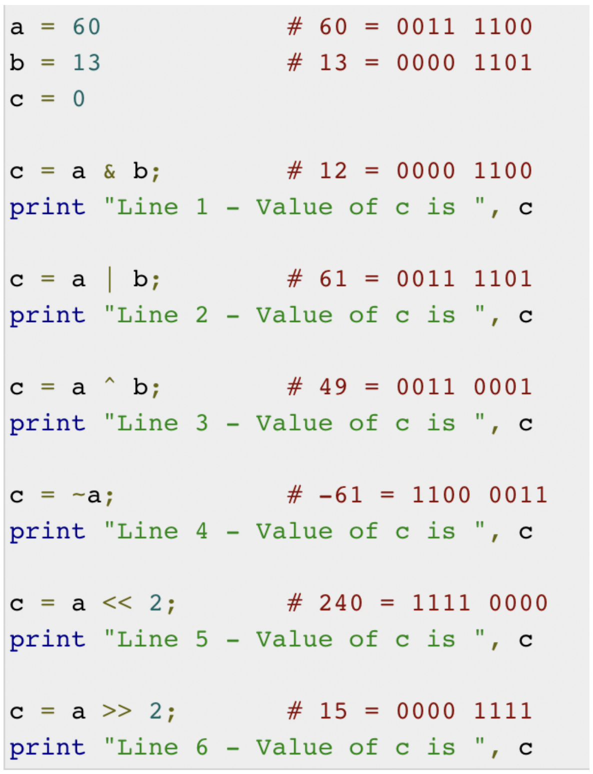
Note: for later version of Python (Not: you must use parentheses '()' for print function)

```
a = 21
b = 10
c = 0

c = a + b
print "Line 1 - Value of c is ", c

c += a
print "Line 2 - Value of c is ", c

c *= a
print "Line 3 - Value of c is ", c

c /= a
print "Line 4 - Value of c is ", c

c  = 2
c %= a
print "Line 5 - Value of c is ", c

c **= a
print "Line 6 - Value of c is ", c

c //= a
print "Line 7 - Value of c is ", c
```

<<fill your result with explanation of each operator>>

20. Bitwise Operators

Note: for later version of Python, you must use parentheses '()' for print function



```
a = 60            # 60 = 0011 1100
b = 13            # 13 = 0000 1101
c = 0

c = a & b;        # 12 = 0000 1100
print "Line 1 - Value of c is ", c

c = a | b;        # 61 = 0011 1101
print "Line 2 - Value of c is ", c

c = a ^ b;        # 49 = 0011 0001
print "Line 3 - Value of c is ", c

c = ~a;           # -61 = 1100 0011
print "Line 4 - Value of c is ", c

c = a << 2;       # 240 = 1111 0000
print "Line 5 - Value of c is ", c

c = a >> 2;       # 15 = 0000 1111
print "Line 6 - Value of c is ", c
```

<<fill your result with explanation of each operator>>

21. Membership Operators

a = 10

b = 20

c = 2

list = [1, 2, 3, 4, 5]

try

a in list

<<fill your result>>

b not in list

<<fill your result>>

c in list

<<fill your result>>