Python notes 1-3

Part 1

What is Python and its history
Python is a high level scripting language designed for highly readable. Python was invented by Guido van Rossum in the Netherlands.

- Python is processed at runtime by the interpreter. You do not need to compile your program before executing it.
- Python is Interactive − You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
- Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
- Python is a great language for the beginner-level programmers

Install Python and install IDE

Python Launcher, IDLE and PyCharm Community Edition

First Python code

```
18
19  print ('hello my name is Tony')
20
```

What are Variable and examples

- Variables are containers for storing data values.
- In Python, variables can contain the same string or value.

```
21  x = 23
22  y = 10
23  print (x)
24  print (y)
25
26  print ('Tony')
27
28  a = 8
29  c = 8
```

Swapping variables

- Variables can be changed

```
19  a = 56
20  b = 29
21  print (a + b)
22  c = 47
23  print (a-c)
24
```

Variable types

- There are total of 5 types of variables

1. Numbers (1,2,66,78…)
2. Strings (‘ and ”)
3. Lists (list = [“car”, “train”, “plane”])
4. Tuples (tuple = [“car”, 7 “train”, 3 “plane”, 1])
5. Dictionaries
    - Dictionaries are used to store data values in key
    - Changeable but not duplicable

Variable arithmetic operators

```
19  a = 56
20  b = 29
21  c = 48
22  d = 10
23
24  print (a+d-b*3)
25
```

Part 2:

If clause

```
a = 56
b = 29
if a > b:
    print ('a is less than b')
    print ('not sure if a is less than b')
```

Result:
a is less than b
not sure if a is less than b

Else

```
c = 5
d = 8
if c > d:
    print ('c is less than d')
else:
    print ('c is not less than d')
```

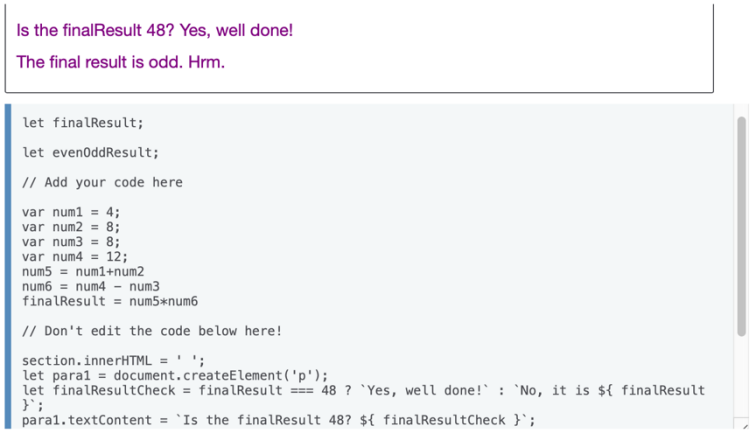
Result:
c is not less than d

Equal to (==)

```
e = 5
f = 8
if e < f:
    print ('e is less than f')
elif e==f:
    print ('e is not less than f')
else:
    print ('e is greater than f')
```

Result:
e is less than f

BMI Calculator



Functions

```
def function():
    print("hello there")
    print("coding is fun")

function()
```

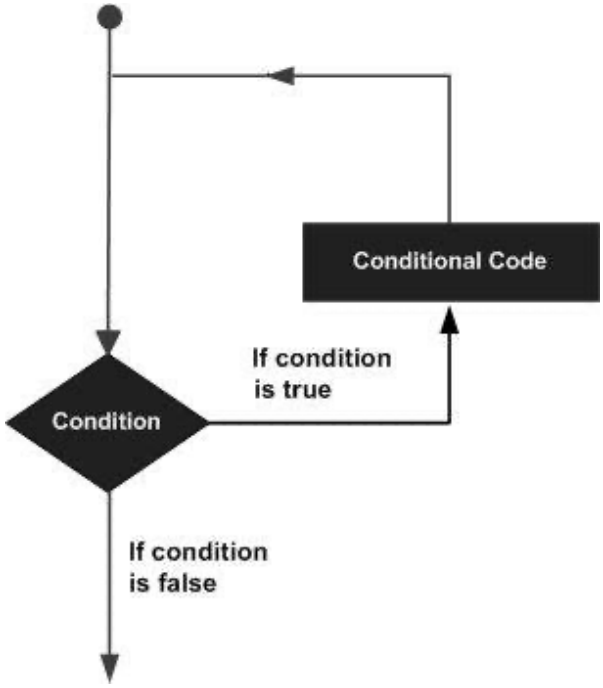
Result:
hello there
coding is fun

BMI Calculator (using functions)

```
name= 'Tony'
height=1.73
weight=55

def BMI():

    bmi = weight / (height * height)

    if bmi < 25:

    print (name)
    print ('is not overweight')

else:

    print (name)
    print ('is overweight')

BMI()
```

Part 3

Loops
A loop statement allows us to execute a statement or group of statements multiple times.



While
Repeats a statement or group of statements while a given condition is true.

For
Executes a sequence of statements multiple times and abbreviates the code that manages the loop variable.

Nested
You can use one or more loop inside any another while, for or do..while loop.

Functions
A function is a block of organized code that is used to perform a single, related action. Functions provide better modularity for your application and a high degree of code reusing.

Creating a function
Function block usually starts with def and then function name and parentheses (**()**).
Any parameters can be put inside the parentheses.
The first statement of a function can be an option.
Every function of the code block begins with a colon (:).

Printme() function
When basic structure of a function is done, the function can be executed by calling it from another function.

Live Demo

```
#!/usr/bin/python

# Function definition is here
def printme( str ):
   "This prints a passed string into this function"
   print str
   return;

# Now you can call printme function
printme("I'm first call to user defined function!")
printme("Again second call to the same function")
```

Keywords argument function
Keyword arguments are related to the function calls. When you use keyword arguments in a function call, the caller identifies the arguments by the parameter name.
This allows you to skip arguments or place them out of order because the Python interpreter is able to use the keywords provided to match the values with parameters.

Lists
Lists are useful when grouping many items into one category.

Creating lists
In Python the items need to be written as a list of comma-separated values between square brackets. ([‘orange’, ‘blue’, ‘yellow’]).

Aceessing values in a list
Use the square brackets for slicing along with the index to gain value available at that index. Always start with number 0 when making a list.

```
25  list_fruits = [ 'apple', 'orange', 'banana'];
26  print 'list_fruits[0]: ', list_fruits[0]
27
```

Basic List operators
Lists respond to the + and * operators much like strings; they mean concatenation and repetition here too, except that the result is a new list, not a string.
In fact, lists respond to all of the general sequence operations we used on strings in the prior chapter.

Python Expression	Results	Description
len([1, 2, 3])	3	Length
[1, 2, 3] + [4, 5, 6]	[1, 2, 3, 4, 5, 6]	Concatenation
['Hi!'] * 4	['Hi!', 'Hi!', 'Hi!', 'Hi!']	Repetition
3 in [1, 2, 3]	True	Membership
for x in [1, 2, 3]: print x,	1 2 3	Iteration

References

https://www.tutorialspoint.com/python/python_loops.htm. Accessed: 4.12.2021

https://www.tutorialspoint.com/python/python_functions.htm. Accessed: 4.12.2021

https://www.w3schools.com/python/python_lists.asp. Accessed: 5.12.2021

https://www.tutorialspoint.com/python/python_lists.htm. Accessed: 5.12.2021