Ethiopian Institute of architecture, building construction and city development, AAU_ EiABC

Programming application short notes: Python Programming Language (part 1 and 2)

Students should revise part one (01_Part one-Introduction to programming and computation), part two (02_Part two_UI-UX) and part three (03_Part three-VisualStudiocode-Github-website design). The good understanding of introduction to programming, visual studio code (code editing/development environment), HTML and CSS areas of the previous components of this course can help students to better understand the following (this) part of the course (04_Part three-python and programming project).

Python part one and part two are the study of Python programming in IDLE (Python shell) and its code/ text editing environment. Furthermore writing and running python programs in windows CMD. [command prompt, help in python, operators, strings, variables, list, tuple, set, data type, number systems, bitwise operators, math functions, etc. are the areas covered while coding in the above mentioned environments. Discussion history of programming language and comparison of python programming language and JAVA programing language is included). In Python part three of the course students will study about developing project in PyCharm.

Students should take a look at all the appendix section of the lecture notes.

Contents

List of figures

List of tables

Programming application short notes: Python Programming language (part 1 and 2)

Python part 1

1. Python programming language introduction

All of the tasks you will be doing on part one and two of your course can be entirely done in an interactive console such as python IDLE. Once you are done with studying part one and two of this course, you can work with PyCharm (IDL) environment for more complex tasks for part three and further components of the course.

1.1. Introduction to python, how to install and the interface

1.1.1. Install:

- Python interpreter (converts your code to bundle), this installation comes with IDLE (interactive console).
- IDE=integrated Development Environment (to work on complex projects: software), software for this course PyCharm developed by jetbrains

Download the software from these links and install them on your computer:

Python: https://www.python.org/downloads/
PyCharm: https://www.jetbrains.com/pycharm/download/#section=windows
While installing PyCharm make sure pip is activated.
PiPy: https://pypi.org/

***Reading online:

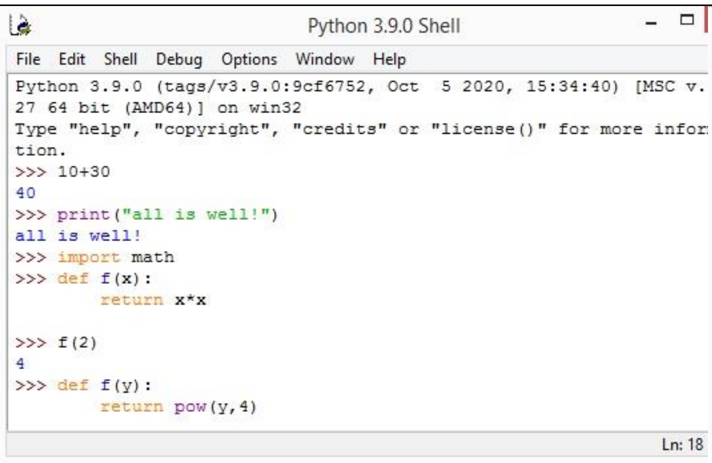
- python docs (google) (

  https://cloud.google.com/python/docs

  https://www.gurobi.com/documentation/9.0/quickstart_mac/py_python_interface.html)

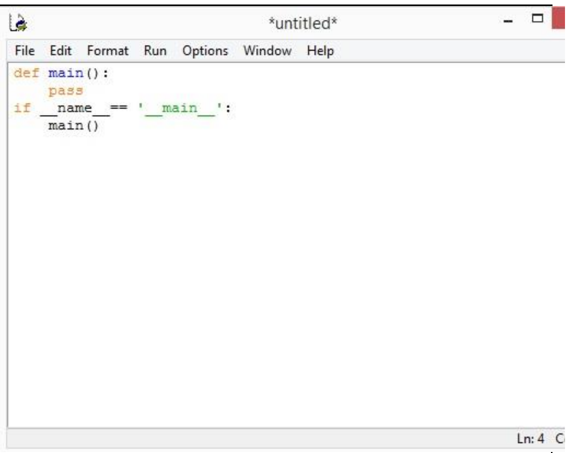
1.1.2. Interface IDLE

IDLE comes built in with python when you first install it. When you open IDLE you will have the python shell window, aka an interpreter window. The interpreter window is used to (1) try out codes or (2) to define commands and do multistep commands. However, the python shell window is not recommended for the latter as it is difficult to work with errors in this window. Editing multi line code is difficult in python shell. The main purpose of python shell is to see the outcome/ output of your programs as you are running them while coding.



```
Python 3.9.0 (tags/v3.9.0:9cf6752, Oct  5 2020, 15:34:40) [MSC v.
27 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more infor
tion.
>>> 10+30
40
>>> print("all is well!")
all is well!
>>> import math
>>> def f(x):
        return x*x

>>> f(2)
4
>>> def f(y):
        return pow(y,4)
```

Figure 1: The python shell / interpreter window



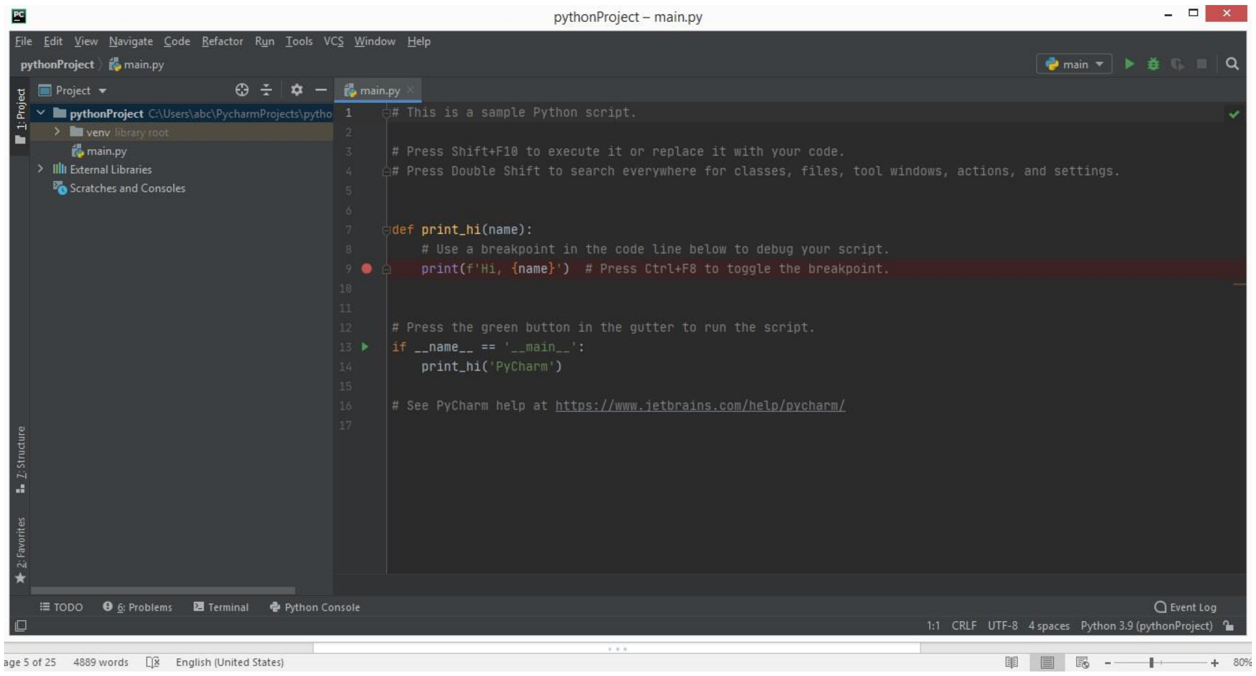
```
def main():
    pass
if __name__== '__main__':
    main()
```

Figure 2: Python code editor window

File + new file on the menu tab open a new editor window or you can open an existing python file via file + open. Once you type in your code in the editor window, to run it the first time you can do so via: run + run module or you can press F5. You will be promoted to save your file and you follow the instruction and save it. Very important while saving your file is to make sure you name your file with .py extension, otherwise the file you will be saving will not be in python. Once you save the file the function will run in your IDLE / interpreter window. Whenever you further edit your program in the editor window and want to run it, you have to press F5 or run + run module and the new program will run in your python shell window.

1.1.3. Interface PyCharm

To open PyCharm go to start + type Pycharm + select Pycharm . When you start PyCharm for the first time, you will have to press on create new project on the resulting window, after which, you will have to indicate location of your project (Default location: C:\Users\UserName\PycharmProjects). Here you can also type in your own project name. Then you will press on create project and you will see the following window:


pythonProject – main.py
File Edit View Navigate Code Refactor Run Tools VCS Window Help
pythonProject 〉 main.py
main
Project
pythonProject C:\Users\abc\PycharmProjects\pytho
venv library root
main.py
External Libraries
Scratches and Consoles
1: Project
Z: Structure
2: Favorites
TODO
6: Problems
Terminal
Python Console
Event Log
1:1 CRLF UTF-8 4 spaces Python 3.9 (pythonProject)
age 5 of 25 4889 words English (United States) 80%

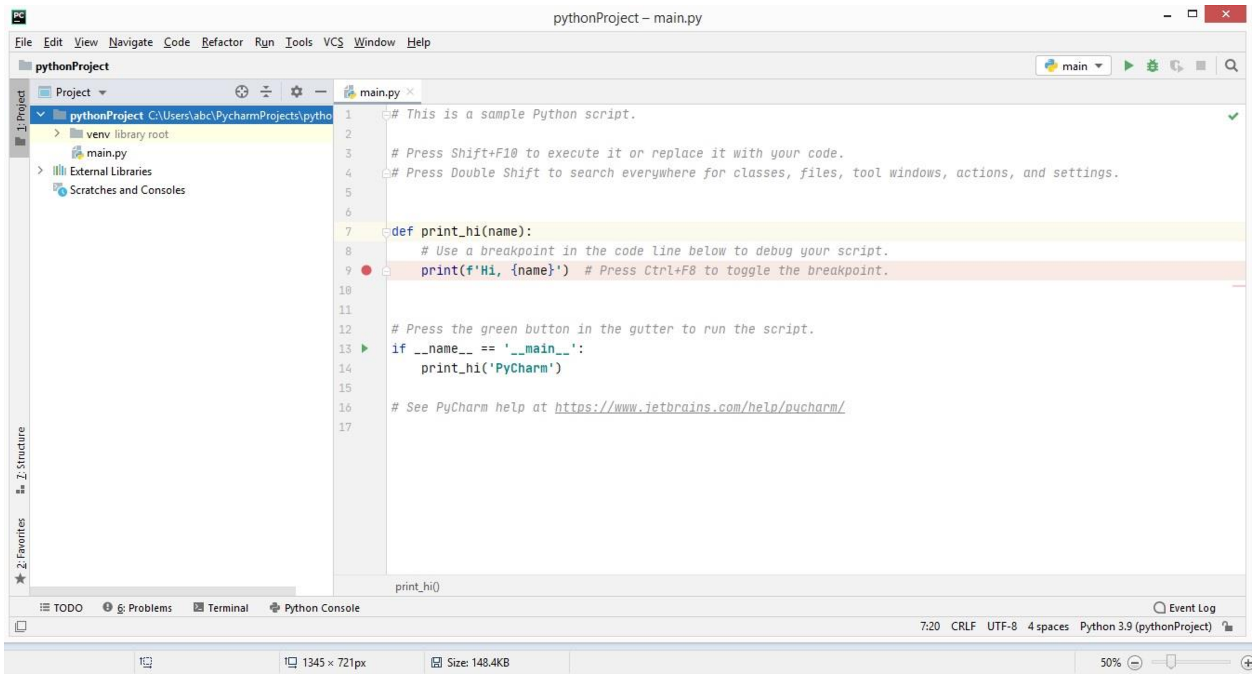
```
# This is a sample Python script.

# Press Shift+F10 to execute it or replace it with your code.
# Press Double Shift to search everywhere for classes, files, tool windows, actions, and settings.


def print_hi(name):
    # Use a breakpoint in the code line below to debug your script.
    print(f'Hi, {name}')  # Press Ctrl+F8 to toggle the breakpoint.


# Press the green button in the gutter to run the script.
if __name__ == '__main__':
    print_hi('PyCharm')

# See PyCharm help at https://www.jetbrains.com/help/pycharm/
```

Figure 3: PyCharm window in Dracula Appearance (Theme)

The left side of the above window is the project browsing section. Functionalities in the interface include: project browser, version control settings, etc. On the most upper part of the window you will find the main menus such as file, edit view, navigate code refactor, etc.   On the right side of your window (text editor) you will write your code and do inspections. On the top right corner, you have a quick access to run and debag code buttons. On the most left lower part of the window you will find quick tools to terminal, python console, run, etc. windows that will popup up on activation.

PyCharm has sidebars for docking tools such as Project structure favorites (on the right), Data base tool on the right and to do, terminal, console etc. on the bottom. This dockings can be shifted around and can be made floating windows. On the very right lower corner we can find the status bar. We can browse through several projects and files in the browser window and we can also open several files that will be indicated in tabs on upper part of the PyCharm window. We can right click on these tabs and sort tabs, rearrange tabs, split screen in to groups, etc. On the left gutter of the editor window you will find functionalities such as: Line numbers, break points, code folding, book marks, etc. The right gutter shows result inspections: Changed lines, warnings, errors and lens mode.

If you would like to change settings of the interface, you can go to menus on the upper part of your window, select file + setting, while here if you change the appearance to IntelliJ Light, your PyCharm window will look like the following:



```python
# This is a sample Python script.

# Press Shift+F10 to execute it or replace it with your code.
# Press Double Shift to search everywhere for classes, files, tool windows, actions, and settings.


def print_hi(name):
    # Use a breakpoint in the code line below to debug your script.
    print(f'Hi, {name}')  # Press Ctrl+F8 to toggle the breakpoint.


# Press the green button in the gutter to run the script.
if __name__ == '__main__':
    print_hi('PyCharm')

# See PyCharm help at https://www.jetbrains.com/help/pycharm/
```

Figure 4: PyPharm window in IntelliJ Light appearance (Theme)

1.2. Python set path in windows and help

1.2.1. Command prompt

You can access your python in your command prompt instead of IDLE.

'CMD + enter' to access your command prompt.

To access python libraries and functions, type in 'python'. If python is not recognized, you can use temporary method by typing in 'set path= 'python' ' however since this solution is not permanent, close your command prompt window and you may follow the following steps to make your access to python through command prompt permanent (unhide hidden folders while carrying out this task):

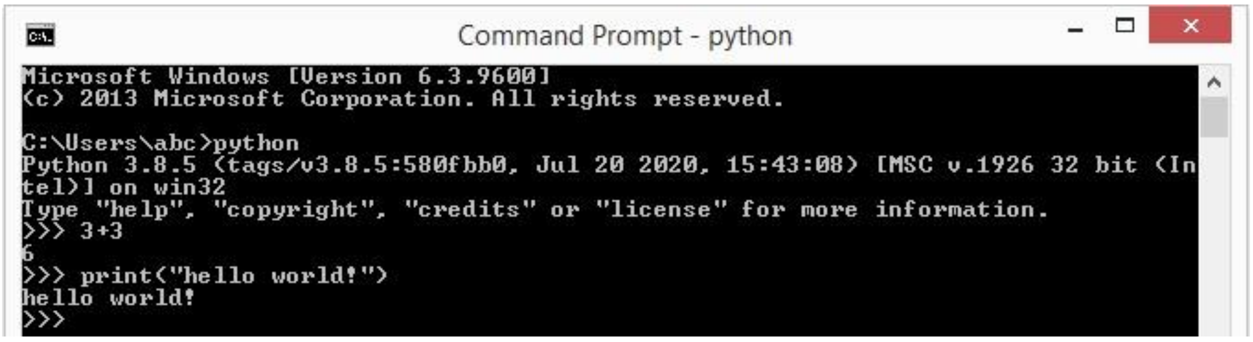
Select the Set to path option while installing python in the first place or

My computer + right click in the window + advanced system setting + environment variables + double click on path (under system variables) + add the paths:

(1) (C:\Users\UserName\AppData\Local\Programs\Python\Python38-32)
(2) (C:\Users\UserName\AppData\Local\Programs\Python\Python38-32\Scripts)

If you will be browsing through these folders, make sure you activate the show/hidden files and folders function of your windows explorer.

Now if you go to your command prompt and type in python, you can access your python functionalities and you can use your command prompt window as development environment just like IDLE.



```
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\Users\abc>python
Python 3.8.5 (tags/v3.8.5:580fbb0, Jul 20 2020, 15:43:08) [MSC v.1926 32 bit (In
tel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> 3+3
6
>>> print("hello world!")
hello world!
>>>
```

Figure 5; Command prompt-Hello world

1.2.2. Help in python

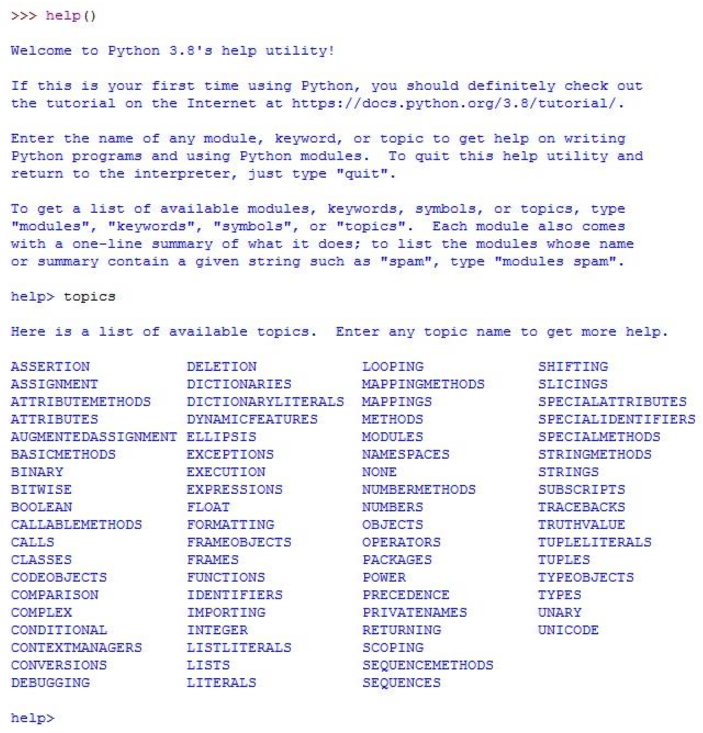
To access the help function of python type 'help()' in your IDLE environment up on pressing enter, you will have access to the help service. You can further type in 'topics' and you will view topics available in your python version. See figure below. You may choose what you want to study further and type the name of the topic and enter, you shall have all information related to that topic for example type in 'set' and press enter.

Table 1: Help topics

help> set Squeezed text (164 lines.) (double click and find more information) help>	help> list Squeezed text (137 lines.) (double click and find more information) help>	help> float Squeezed text (200 lines.) (double click and find more information) help>

To go back to your command function from help type in 'quit'

You can also just type in 'help (list)' and get access to help available with regards to a specific topic.



```
>>> help()

Welcome to Python 3.8's help utility!

If this is your first time using Python, you should definitely check out
the tutorial on the Internet at https://docs.python.org/3.8/tutorial/.

Enter the name of any module, keyword, or topic to get help on writing
Python programs and using Python modules.  To quit this help utility and
return to the interpreter, just type "quit".

To get a list of available modules, keywords, symbols, or topics, type
"modules", "keywords", "symbols", or "topics".  Each module also comes
with a one-line summary of what it does; to list the modules whose name
or summary contain a given string such as "spam", type "modules spam".

help> topics

Here is a list of available topics.  Enter any topic name to get more help.

ASSERTION           DELETION            LOOPING             SHIFTING
ASSIGNMENT          DICTIONARIES        MAPPINGMETHODS      SLICINGS
ATTRIBUTEMETHODS    DICTIONARYLITERALS  MAPPINGS            SPECIALATTRIBUTES
ATTRIBUTES          DYNAMICFEATURES     METHODS             SPECIALIDENTIFIERS
AUGMENTEDASSIGNMENT ELLIPSIS            MODULES             SPECIALMETHODS
BASICMETHODS        EXCEPTIONS          NAMESPACES          STRINGMETHODS
BINARY              EXECUTION           NONE                STRINGS
BITWISE             EXPRESSIONS         NUMBERMETHODS       SUBSCRIPTS
BOOLEAN             FLOAT               NUMBERS             TRACEBACKS
CALLABLEMETHODS     FORMATTING          OBJECTS             TRUTHVALUE
CALLS               FRAMEOBJECTS        OPERATORS           TUPLELITERALS
CLASSES             FRAMES              PACKAGES            TUPLES
CODEOBJECTS         FUNCTIONS           POWER               TYPEOBJECTS
COMPARISON          IDENTIFIERS         PRECEDENCE          TYPES
COMPLEX             IMPORTING           PRIVATENAMES        UNARY
CONDITIONAL         INTEGER             RETURNING           UNICODE
CONTEXTMANAGERS     LISTLITERALS        SCOPING
CONVERSIONS         LISTS               SEQUENCEMETHODS
DEBUGGING           LITERALS            SEQUENCES

help>
```

Figure 6; Help

Table 2: IDLE, IDLE's text editor and PyCharm and its console

Demonstration (IDLE, IDLE's text editor and PyCharm and its console). Execute all four tasks indicated below as an indication of your understanding of all python editing and development environment and your understanding of the interfaces related.		
(1) Type and run the function below in IDLE. Then retype the same function in ILDE's text editor and save it in .py format on your computer and then run it in IDLE.	(2) Type and run the function below in PyCharm and run it in PyCharm's console.	(3) Type and run the function below in window's command prompt and run it in windows command prompt (cmd)

x=3
y=10
z=x+y
print(z)		

(4) Browse to the saved .py file of the function below via windows command prompt and run the file in windows command prompt.		

1.2.3. IDLE previous command and clear screen

Options + configure IDLE + keys + scroll in action keys and find history of previous key + select it and change the default key to your liking + get new keys for selection + …

Change the keys for restarting your IDLE environment, your history key, etc. And show it to your instructor.

In windows, we can clear our command by typing: cls + enter (you clear your screen). In IDLE it is not possible to clear your window. You can only scroll through the screen. But you can restart IDLE from any line you are at (Navin Reddy). To do so, go to shell menu: shell+ restart shell.

1.3. Programming languages

You have already discussed this topic in an earlier component of your course ‘introduction to programming’. The following text is to summarize the mentioned lesson and add specific information about python programming language.

Binary is to programing languages as English or Amharic is to speech. In your endeavors of using computers, programming languages can be understood as intermediate communication that lay between binary and languages such as English or Amharic.

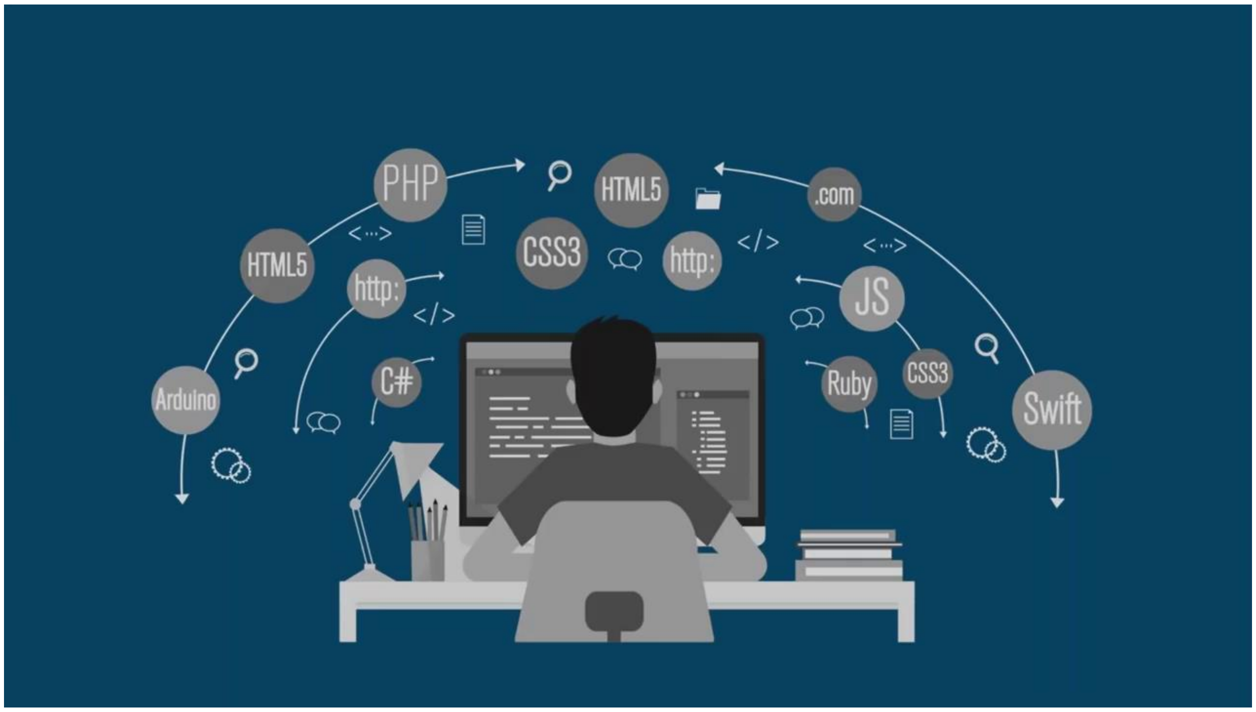


Figure 7: Programming languages

1.3.1. Comparison of programing languages:

You can compare programing languages by making use of questions such as the following:
(Is it Compiled or interpreted?)
(Is it Strong or weak typing?)
(Does it have Garbage collection functionality?)
(Is it object oriented programming?)
(Is it compiled or interpreted?)

1.3.2. Discuss about programming styles:

(1) Imperative programming: is just list of instructions without blocks and loops.
(2) Structural Programming: based on imperative programing uses: **if blocks, loops**
(3) Procedural Programming has **if blocks, loops, functions**
(4) Object oriented programming: uses: **if blocks, loops, functions, objects**
(5) Declarative programming (we tell the program what we want instead of how to do a task)
(6) Functional programming (nothing happens outside of the scope of the functions used. Since it is declarative, we do not use loops. To replace loops we use recursions.)

1.3.3. History of python programing language

Python is very popular for its simple programming syntax, easy code readability and English like command making coding easy and efficient. Python is interpreted, high-level and general purpose programming language. The programming language is developed by Guido Van Rossum (https://en.wikipedia.org/wiki/Guido_van_Rossum) at the National Research Institute of Mathematics, computer science in Netherlands during late 1980's. Python is named after the show 'Monty Python's flying Circus'. The language was developed as a successor of ABC programing language. According to him Python programing language is designed to have the benefits of both shell and C programming languages. Version 1.0 of python were released in 1994 with features such as Exception handling, lambda, map, and filter and reduce. In 2000, python 2.0 was released introducing features like List comprehensions and garbage collection systems capable of collecting reference cycles. 2008 python 3.0 was released. Python 2.x and python 3.x are the most used versions. Python 3.9 version is the currently available stable version of python. Python is mostly used for developing, scripting and software developing.

Companies such as Google, Facebook, Instagram, Spotify and Netflix use python. Applications of python are seen in the areas of Machine learning, artificial intelligence, data science and IoT. Python provides libraries such as: SciPy, Keras, NumPy, Flask, Django and TensorFlow. Pep-8 is the official guide that defines the styling conventions for Python code. Popular projects by python include: PipENV, Chatistics, self-organizing maps and python t BPF converter. Some of the trending Python repositories on Github today include, NLP, ML-from-scratch, PyTorch-transformers, 100-Days of ML Code and ETC. websites such as Redit and Quora are built using python. Games developed by python include: Civilization IV, Battlefield 2, Sims 4, and World of Tanks. Users and admirers that are knowledgeable are referred to as Pythonists, Pythonistas, and Pythoneers (simply learn).

1.3.4. Comparison between java and python

Figure 8, Python and Java

Programming language Name	Python Inspired by the comedy Movie: Monty Python's Flying Circus	Initially called 'Oak' after an oak tree and then named 'Green' and then finally renamed 'Java', from Java coffee, of Indonesia
Developer	Guido Van Rossum	Sun Microsystems, Inc., Led by James Gosling
Year	1989	1995
Definition	Python is a simple, open source and object oriented programming language used for artificial intelligence, machine learning, web development and many other applications. Python is a general purpose language. It is used to do almost anything (game development, Ai, scientific computation, data analysis, machine learning, etc.).	Java is an object oriented and platform or machine independent programming language used for developing various applications such as mobile applications and websites.
Speed	Python is dynamically typed (type checking is done at run time) and interpreted language as such it is relatively slow. The speed is determined by the type of data and the variable data type at run time.	As java is statically typed (type checking is done at compiling) and compiled language it is relatively faster.

Code and syntax	Lines of code:	Less lines of code than java.
		Example:
		`>>> print("hello world")`
			More lines of code than python.
			`public class Sample`
			`{`
			`  public static void main(string[] args)`
			`  {`
			`    // printing a statement in java`
			
			`    System.out.println("hello world");`
			`  }`
			`}`
			Example:

	Variable declaration and semicolon	`>>> x=10`
		`>>> y=20`
		`>>> print(x+y)`
			`public class Sample`
			`{`
			`publicstatic void main (string[] args)`
			`{// Sum of two numbers in Java`
			`  int x=25;`
			`  int y=15;`
			`  int sum=x+y;`
			`  System.out.print(sum);`
			` }`
			`}`

	Indentation	Indentation is a must in python. There will be an error result if you miss indentation. While coding in PyCharm, you will notice that after every statement you will commonly use four spaces while other number of spaces are allowed. However the number of spaces you are using should always be the same.	Indentation is optional. But programmers use indentation to make their codes readable and logical for other programmers. Missing indentation will not result in errors. Perhaps since you have studied CSS on early part of your course you clearly understand with this means.

	Classes and objects	`>>> class Employee:`
		`            def __init__(self,name,game):`
		`                self.name=name`
		`                self.game=game`
		`            def show(self):`
		`                print(self.name+""+self.game)`
		`      Player=new Player("Kebede","Goal Keep`
		`      Player.show()`
			`public class Player`
			`{ string name;`
			` int game;`
			` Player(string p, int g)`
			`  {`
			`   name = p;`
			`   game = g;`
			`  }`

			void show()
			{
			System.out.println(name + "" + game);
			}
			public static void main(string[] args)
			{
			Player e1 = new
			Player ("Kebede", Goal Keeper);
			}
			}

Practical agility		Used for AI, IOT, machine learning, scientific programming, etc.	Used for mobile and web application
Ease of learning		More user friendly and intuitive coding style. Less complex.	Relatively less friendly for coding and relatively more complex for beginners.
Trend		Trend of use of Python is increasing. Python is becoming the most popular language to introduce students to programming. Python is popular among programmers with work experience of 6 years and less. As such salary for python among this group is higher than java.	Trend of use of java is decreasing. Java is more popular among programmers with long years of work experience. (Programmers 6 years and above. As such salary for java among this group is higher than python as python came in to use latter than python.)

Figure 9: Python and java comparison. Source: (Placeholder1)

1.4. Operators in python: part one

1.4.1. Operators

+ (addition), - (subtraction), *(multiplication), / (division with float results), // (division with integer results), () (parenthesis), ** (exponents), % (modulus, with a remainder result of division of two numbers).

Table 3: Operations

>>> 1+10
11
>>> 1-10
-9
>>> 1*10
10
	>>> 1/10
	0.1
	>>> 1//10
	0
	>>> 1%10
	1
	>>> 2**3
	8
		>>> 1+2*8
		17
		>>> 1+2*8-3
		14
		>>> 1+(2*8)-3
		14
		>>> (1+2)*8-3
		21
		>>> (1+2)*(8-3)
		15

Exercise:

Make use of your IDLE environment and compute the following problem.

$$Y=a(r+1)^t$$

The above formula is to a value that exponential growth over time. 'Y' value after 't' period of time, 't' is the total period of time. 'a' is initial value , 'r' is rate at which a value increases during the period 't'. If initial salary of a person is 5000 birr and increases by 6% every year, what is the value of Y in 10years?

Answer:

[redacted]

Rearrange the following operations in their hierarchical order first to last (division, multiplication, subtraction, exponents, and addition).

Answer:

[redacted]

There are operators for arithmetic, relational, logical and bitwise operations. Try to Rearrange the following operators in their hierarchical order first to last, based on your past experience.

logical OR (||) or (or), multiplicative (*, / ,%) , postfix and unary (++ , -- , !) , logical AND (&&) or (and), Parenthesis (()) , additive (+ , -) , assignment (= , += , -=, *=, /=, %=) , relational (< >, <= ,>=), equality (== !=)

Answer:

[redacted]

1.5. Strings

' ' (single quotes), " " (double quotes) : discuss the difference between the two and how to use them in different combinations.

To avoid interpretation of single or double quote in our strings we can type \ (backward slash) in front of the quote we want to avoid interpreting.

You can concatenate strings: ‘text’+’text’ , 10*’text’

print(‘text’) ,

you can use the raw string syntax to print a string as it is in case of undesired meaning of built in syntaxes by simply typing r in front of the ‘text’

\n in python means new line.

Table 4: Strings

>>> print('all is well')
all is well
 	
>>> print("all is well")
all is well
 	
>>> print("it's all good")
it's all good
 	
>>> print('all is "well"')
all is all "well"
	>>> 'all is well' + ' if you are happy'
	'all is well if you are happy'
	
	>>> 10*'Things get better. '
	'Things get better. Things get better. Things get better. Things get better. Things get better. Things get better. Things get better. Things get better. Things get better. Things get better. '
	
	>>> print('it\'s all "well"')
	it's all "well"
	
	>>> print('All is well. \nThings will get better.')
	All is well.
	Things will get better.
	
	>>> print(r'All is well. \nThings will get better.')
	All is well. \nThings will get better.
		Print the following text all in one line in your IDL environment (take note the fact that ‘\n’ is printed as new line in python. Also take note of the fact that putting ‘r’ in front of any string allows you to print it as it is without further consideration of syntax.):
		
		Sara’s file is located in the computer’s hard disk, C:\new-folder1\new-folder2.
		
		Answer:
		
		>>> [redacted]
		
		>>> [redacted]
		
		[redacted]

1.6. Variables in python

1.6.1. Variables - Part 1

Variable is a container in which we add values. For example in x=3, x is the variable and 3 is the value we stored in a variable called x. in other words we are assigning 3 to x.

***Discuss about the difference between java and python when assigning values to variables.

Under score (_) prints the output of the previous operation, you can also see what other keys do by going to ‘option + configure IDLE + keys’ . # in front of any line of code makes the line not to be processed by python. You can also use it to add comments in your program. You might remember that /* comment */ for CSS and < !--Comment -- > for HTML

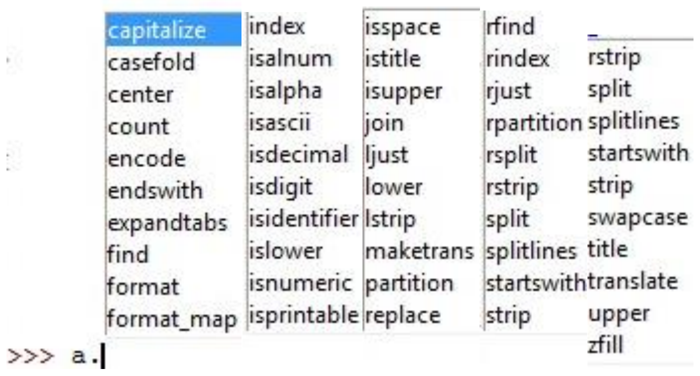
You can assign string values to variables. For example x=’person’.
We can fetch letters from a letter values from a string by making use of the index value []. For example: x[0] is ‘p’ and x[4] is o. while fetching letter values form a string , you can use 0 to as many positive integers from left to right or -1 to as many negative integers from right to left. For example: x [-1] is s. by separating two numbers with colon in the rectangular bracket, you can print more than one string (range of strings). For example x[0:3] prints ‘ad’. If we place a valid number and followed by a colon, we will print all letters starting from the letter value of the number and all letters after it.

To compute length of a sting we can use the function len(variable). For example: x=’all is well’, len(x) will be 11. White space in a string has an index.

Index	0	1	2	3	4	5
String =	p	e	r	s	o	n
index	-6	-5	-4	-3	-2	-1

Figure 10, Index of letters in a string

Table 5: Variables 1

>>> x=10
>>> y=3
>>> x+x**y
1010
 	
>>> a='people '
>>> b='of '
>>> c='earth.'
>>> print(a+b+c)
people of earth.
 	
>>> a.count('e')
2
>>> a.replace('p','t')
'teotle '
	X=’person’
	>>> print(x[0])
	P
	>>> print(x[-6])
	p
	>>> print(x[0:3])
	Per
	>>> print(x[1:])
	erson
	>>> print(x[:4])
	pers
	>>>
	z=’person’
	>>> len(z)
	6
	Strings in python are Immutable.
		Reading Assignment:
		Study the different possible functions (string variables) in strings. Such as: capitalize, casefold, center, count, encode, endswith, expandtabs, find,ETC.
		 capitalize casefold center count encode endswith expandtabs find format format_map index isalnum isalpha isascii isdecimal isdigit isidentifier islower isnumeric isprintable isspace istitle isupper join ljust lower lstrip maketrans partition replace rfind rindex rjust rpartition rsplit rstrip split splitlines startswith strip rstrip split splitlines startswith strip swapcase title translate upper zfill
		>>> a.
		After typing dot next to your string variable press ctrl+ spacebar and you will have the above functions as a pop up.

1.6.2. Variables - Part 2

In python if two variables have the same data they will have the same address or ID making python's memory efficient. Two or more variables referring to the same value have the same address or ID. However, we can change the values of variables changing their address or ID while keeping the name of the variable.

Address of an object is not based on the name of the variable but the data or value itself.

Variables are also called tags. Once you are using different values exchanging and changing their values, the values that are not being used or tagged with any variable will be taken care of by the garbage collection function of python programming. In python, we cannot change variables to constant but we can show our intention by typing the variable name in caps locks. Variables have inbuilt types such as floats, integers, etc. in order to identify the type of variable you can type in 'type (variable)'. We can also create our own variable types.

Table 6: Variables 2

>>> a='all is well.'
>>> a
'all is well.'
>>> id(a)
45974576
>>> x=10
>>> x
10
>>> id(x)
1876076608
>>> y=5
>>> id(y)
1876076528
	>>> z=y
	>>> z
	5
	>>> id(z)
	1876076528
	>>> id(5)
	1876076528
	>>> x=y
	>>> x
	5
	>>> id(x)
	1876076528
		Constants:
		
		>>> GRAVITY=9.8
		>>> type(GARAVITY)
		<class 'float'>
		
		>>> PI=3.14
		>>> type(PI)
		<class 'float'>

1.7. List in python

List is a group of elements in rectangular bracket. For example x=[10,20,60,87] is a list and also y=['lion', 'dog', 'car', 'house']. Furthermore z=[2.3, 'dog',25, 'person'].

In order to fetch different elements of these lists, we can use the same principles we used above for strings through index values. For example: x[0] is 10, x[0:3] is [10,20,60]. Y[-1] is ['house'], z[0:3] is [2.3,'dog',25]

We can work with all of the above lists with each other and create list of lists.

For example: a=[x,y,z] is [[10,20,60,87], [‘lion’, ‘dog’, ‘car’, ‘house’], [2.3, ‘dog’,25, ‘person’]]

We can perform operation on the lists through functions. By typing dot (.) after name of the list we can have different built in functions made available to us such as: append, clear, copy, count, extend, index, etc.

To add an element in a list we use append, insert, etc. For example: x.append(65) will change the list x to [10,20,60,87,65]. And x.insert(0,100) will change the list x to [100,10,20,60,87]. X.remove(20) will change the list x to [10,60,87]. We can also remove an element from a list: x.pop(3) will change the list x to [10,20,60] by removing 87. We can also delete an element through index. For example: del x[2:] will change the list x to [10,20] by deleting all elements expect elements 0 and 1 indexes.

Table 7: List

List is mutable, meaning values can be changed.
 	
x=[10,20,60,87]
x.clear()
x=[]
 	
Extend lists by multiple values: mind the application of both curve and rectangular brackets
c=[10,15,20,25]
c.extend([30,35,40])
c=[10, 15, 20, 25, 30, 35, 40]
	x=[10,20,60,87], y=[‘lion’, ‘dog’, ‘car’, ‘house’], z=[2.3, ‘dog’,25, ‘person’
	list of lists:
	a=[x,y,z]
	a= [[10,20,60,87], [‘lion’, ‘dog’, ‘car’, ‘house’], [2.3, ‘dog’,25, ‘person’]]
		x=[10,20,60,87]
		
		x.append(65)
		x =[10,20,60,87,65]
		x.insert(0,100)
		x =[100,10,20,60,87]
		X.remove(20)
		X= [10,60,87]
		x.pop(3)
		x =[10,20,60]

	Inbuilt functions such as max, sum, min and etc.:
	x=[5,20,35,7,12]
	min(x)=5
	max(x)=35
	sum(x)=79
		sort lists:
		x.sort()
		x=[5, 7, 12, 20, 35]

**Mutable (we can change value of an element) and immutable (we cannot change value of an element)		

1.8. Tuple and set in python

Tuple is similar to list except that in list we can change values while in tuple we cannot change values of elements. Meaning, list is mutable where tuple is immutable. We cannot perform append, remove, etc. in tuple. We can only count or index. Since the values in tuple do not change, iteration operations are faster in tuple than list. Unlike list it uses curve brackets ().

Sets use curly bracket, list uses rectangle bracket and tuple uses curve brackets. Set is a collection of unique elements. Set is mutable we can change values. Set does not follow sequence. It uses hash function, as a result, in stets we cannot have index function for the elements unlike list and tuple.

Index	0	1	2	3	4	5
Tuple =	(20	23	42	10	50	45)
index	-6	-5	-4	-3	-2	-1

Figure 11: Index of elements in a tuple

Table 8: Tuples and sets}

Tuple:
t=(20,23,42,10,50,45)
We can fetch an element with index (mind the use of the rectangle bracket)
t[2]=42
 	
>>> t.count(42)
1
>>> t.index(42,1,4)
2
>>> t.index(10)
3
 	
>>> len(t)
6
 	
Tuple unlike sets and lists is immutable.
	Sets:
	s={12,12,78,7,7,1}
	>>> s
	{1, 12, 78, 7}
	
	Since we do not have proper sequence in set, index is not supported in sets.
	
	>>> s={20,23,42,10,50,45}
	>>> len(s)
	6
	
	>>> r={10,10,20,30,30}
	>>> len(r)
	3
	
	Sets like list are mutable. Sets do not support duplicate values.
		s={12,4,85,15}
		>>> s
		{85, 12, 4, 15}
		>>> s.add(100)
		>>> s
		{100, 4, 12, 15, 85}
		>>> s.remove(85)
		>>> s
		{100, 4, 12, 15}
		
		>>> r={15,12,500,45}
		>>> r
		{45, 12, 500, 15}
		>>> s.union(r)
		{100, 4, 12, 45, 15, 500}
		>>>
		>>> s.intersection(r)
		{12, 15}
		>>>

>>> r={10,20,30,60}
>>> s={30,60,70,80}
>>> t={60,80,100,200}
 	
Compute the intersection of all the above three sets in your IDLE environment: r n s n t
 	
Answer:
[redacted]
 	
{60}
 	
Compute the union of all the above three sets in your IDLE environment: r u s u t
 	
Answer:
[redacted]		

{100, 200, 80, 20, 60, 10, 30}

1.9. Data types in python

1.9.1. None:

When a variable is not assigned to any value it is called none. In other language we can use the key word null.

1.9.2. Numeric:

int, float, complex, bool

Table 9: Numeric Data Type

>>> y=3
>>> x=3.6
>>> type(x)
<class 'float'>
>>> y=6
>>> type(y)
<class 'int'>
>>> z=2+8j
>>> type(z)
<class 'complex'>
	>>> x=3.6
	>>> y=int(x)
	>>> y
	3
	
	>>> type(y)
	<class 'int'>
	>>> z=float(y)
	>>> z
	3.0
	>>> c=complex(x,z)
	>>> c
	(3.6+3j)
	>>> type(c)
	<class 'complex'>
		>>> y=3
		>>> x>y
		True
		>>> x<y
		False
		>>> bool=x<y
		>>> bool
		False
		>>> int(true)
		>>> int(True)
		1
		>>> int(False)
		0	

1.9.3. Sequence data types:

- list, tuple, set, string, range

Table 10: Sequence Data type

>>> l=[10,50,11,2,3]
>>> type(l)
<class 'list'>
>>> s={10,20,11,2,3}
>>> type(s)
<class 'set'>
>>> t=(10,20,11,2,3)
	>>> s="all is well."
	>>> type(s)
	<class 'str'>
	>>> t='g'
	>>> type(t)
	<class 'str'>
		>>> range(12)
		range(0, 12)
		>>> list(range(12))
		[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]
		>>> list(range(2,12,2))
		[2, 4, 6, 8, 10]	

>>> type(t) <class 'tuple'>		>>> list(range(1,12,3)) [1, 4, 7, 10]	

1.9.4. Dictionary data types (Map)

If we have a large data and we want to fetch selected data efficiently, we can assign a key to the data and fetch accordingly. In list, we do the same task with index but in dictionary (map) we use keys. Keys have to be unique for every data. We use curly bracket for dictionaries since we do not want to repeat values. Since sets do not support repeated values, working with sets is a proper way.

Table 11: Dictionary

>>> x={'sara':'door', 'bekele':'window', 'chala':'roof'}
>>> x.keys()
dict_keys(['sara', 'bekele', 'chala'])
>>> x.values()
dict_values(['door', 'window', 'roof'])
>>> x['bekele']
'window'
>>> x.get('sara')
'door'	

1.10. Operators in python part 2

Parenthesis (()), postfix and unary (++, -- ,!) , multiplicative (* , /, //, %), additive (+ ,-), relational (<, > ,<=, >=), equality (== , !=) , logical AND (&&) or (and) , logical OR (||) or (or), assignment (=, +=, -=, *=,/=, %=)

1.10.1. Arithmetic operators - multiplicative (*, /, //, %), additive (+, -)

****Refer to operators in python Part 1

+ (addition), - (subtraction), *(multiplication), / (division with float results), // (division with integer results), () (parenthesis), ** (exponents), % (modulus, with a remainder result of division of two numbers).

1.10.2. Assignment operators (= , += , -= , *=, /=, %=)

Table 12: Operators 1

>>> x=21	>>> a=30	We can assign value to multiple variables
>>> y=13	>>> b=15	once instead of assigning values one at a
>>> x=x+2	>>> a*b	time.
>>> x	450	>>> y,z=8,10
23	>>> a/b	>>> y
>>> x+=2	2.0	8
>>> x	>>> a//b	>>> z
25	2	10
>>> x*=2	>>> a%b	>>> x+z
>>> x>>50	0	>>18
>>> a=10	X=50	
>>> y=5	>>> x-=2	
>>> a=10	>>> x	
>>> b=5	48	
>>> a-b	>>> y=30	
5	>>> y/=3	
>>> a+b	>>> y>>10.0	
15	>>> y=30	
	>>> y%=3	
	>>> y>>0	

1.10.3. Relational operators - (< , > , <= , >=)

1.10.4. Logical operators - equality (== , !=) , logical AND (and), logical OR (or),

1.10.5. Postfix and Unary operators (-,+,--, ++, not)

Table 13; Logical and logical or

Logical ‘and’ and ‘or’ operation			
Variable	variable	value	value
x	y	Boolean and	Boolean or
0 (false)	0(false)	0 (false)	0 (false)
0 (false)	1 (true)	0 (false)	1 (true)
1 (true)	0 (false)	0 (false)	1 (true)
1 (true)	1 (true)	1 (true)	1 (true)

Table 14: Operators 2

>>> x=10	>>> x=10	>>> c=12	>>> x=5
>>> y=5	>>> y=5	>>> -c	>>> y=3
>>> x>y	>>> x<12 and y<6	-12	>>> x>1 or y<6
True	True	>>> +c	True
>>> x<y	>>> x<6 and y<6	12	>>> not x
False	False	>>> --c	False

>>> x==y
False
>>> y=10
>>> y==x
True
>>> x>=y
True
>>> x<=y
True
>>> x!=y
False
	>>> x<6 or y<6
	True
	>>> x=True
	>>> x
	True
		12
		>>> ++c
		12
		>>>	

1.11. Number system conversion in python

In programing world we use binary (is in base 2 and we u0se 0 and 1, BIT (binary digit)) and decimal (is in base 10 and we can go from 0 to 9) number system. Furthermore we also use Octal (is in base 8 and we can go from 0 to 7) and HexaDecimal (is in base 16 and it goes from 0 to 9 and after 9, it goes from a to f). In your IDLE environment, binary numbers are displayed with a prefix of 0b, octal numbers with a prefix of 0o and HexaDecimal numbers with prefix of 0x. To convert a decimal number (6) to binary, octal and hexadecimal, you can follow the following rules consecutively: bin(6), oct(6) and hex(6), in each case press enter after you typed them in your IDLE environment. If you want to convert a binary number to decimal number in your IDLE environment, type the binary number by adding the prefix 0b and press enter. For example to find the decimal number value of the binary number 101110, enter 0b101110 to your IDLE and press enter, you will find 46 as the decimal number value. For octal to decimal, add the prefix 0o to the octal number and press enter in your IDLE environment. For HexaDecimal to decimal, add the 0x prefix to your hexadecimal number and press enter in your IDLE environment.

Table 15: number system

>>> bin(46)
'0b101110'
>>> 0b0101
5
	>>> oct(47)
	'0o57'
	>>> 0o57
	47
		>>> hex(47)
		'0x2f'
		>>> 0x2f
		47

Class exercise: a. By making use of IDL Convert the decimal number 123 to binary, octal and HexaDecimal. Answer (1111011 , 173 and 7b respectively) b. By making use of pen and paper convert the decimal number 35 to binary. Answer (100011), see table below. c. By making use of pen and paper convert the binary number 1100100 to decimal number. Answer (100), See table below. d. By making use of pen and paper convert the decimal number 35 to octal number value. e. By making use of pen and paper convert the octal number 43 to decimal number.		

Answer b:

Table 16: decimal number to binary conversion

			Remainders	
2	/	35		
2	/	17	1	Λ
2	/	8	1	Λ
2	/	4	0	Λ
2	/	2	0	Λ
		1	0	Λ
		>	>	

Collect the remainders as shown on the table, the binary number value of the given decimal number is: 100011

Answer c:

Table 17: Binary to decimal number conversion

1	1	0	0	1	0	0
$1 * 2^6$	$1 * 2^5$	$0 * 2^4$	$0 * 2^3$	$1 * 2^2$	$0 * 2^1$	$0 * 2^0$
64	32	0	0	4	0	0

The decimal number value of the given binary number is 64+32+0+0+4+0+0 = 100

Answer d:

Table 18: decimal to octal conversion

			Remainders	
8	/	35		
		4	3	Λ
		>		

Collect the remainders as indicated on the table, the octal number value of the given decimal number is: 43

Answer e:

Table 19: octal to decimal conversion

4	3
$4 * 8^1$	$3 * 8^0$
32	3

The decimal number value of the given octal number is: 32+3 = 35

1.12. Python bitwise operators

There are six types of bitwise operators:

(1)Complement (~) or tilde operator ; (2) And(&) ; (3) Or(|) ; (4) XOR(^) ; (5) Left Shift(<<) ; (6) Right Shift(>>)

1.12.1. The Complement bitwise operator

To solve bitwise complement of decimal numbers, first convert the number to binary format, after that add one to the last digit of the binary number. And then change the resulting binary number to decimal and postfix it as a negative value or multiply it by -1. For example if we want to find the bitwise complement of 14, we will first change it to its binary format which is 1110 and we add 1 to it changing it to 1111. Now all we have to do is change the resulting binary number to decimal number and postfix it or multiply it by negative one. To find ~25, change to binary =11001, add 1=11010. Change 11010 to decimal = 26 and then multiply the number by -1 = -26.

***Read about two's complement method for further understanding.

1.12.2. The And (&) bitwise operator

Here unlike logical AND (and) we use the ‘ &’ symbol. To find the bitwise and (&) value of two decimal numbers (see table below) convert each to their binary value and apply the logical AND (and) operator between each corresponding numbers. Change the resulting binary number to decimal value and that will be your answer. For example to find 15&14,

Table 20: bitwise and (&)

15	1	1	1	1
14	1	1	1	0
15 and 14 (logical and)	1	1	1	0

Class exercise, by making use of pen and paper work out and show 50 & 10

1.12.3. The OR (|) bitwise operator

Here unlike logical OR (or) we use the ‘|’ symbol. To find the bitwise and (I) value of two decimal numbers (see table below) convert each to their binary value and apply the logical OR(or) operator between each corresponding numbers. Change the resulting binary number to decimal value and that will be your answer. For example to find 15 | 14,

Table 21: Bitwise or (|)

15	1	1	1	1
14	1	1	1	0
15 or 14 (logical and)	1	1	1	1

Change the binary number 1111 to decimal = 15

Class exercise, by making use of pen and paper work out and show 50 | 10

1.12.4. The XOR (^) bitwise operator

Here unlike logical OR (or) or the bitwise or (|) we use the cap symbol (^). To compute this operation we change the value of decimal numbers to binary numbers and we compare pairs of the corresponding numbers in the binary numbers. In case of even or zero number of 1 (if both numbers are similar that is either 0 and 0 or 1and 1), we make out 0. In case of odd number of 1 (if 1 and 0 or 0and 1), we make out 1.

See the following example:

To work out 15^10 we first convert the numbers to binary: bin(15)= 1111 and bin(10)= 1010

Table 22: Bit wise XOR (^)

15	**1**	**1**	**1**	**1**
10	**1**	**0**	**1**	**0**
15^10	**0**	**1**	**0**	**1**

Change the binary number 0101 to decimal place: 0b0101=5

1.12.5. Left shit (<<) bitwise operator

To shift a decimal number to left by a certain number means to convert the decimal number given to binary and then move the decimal point of the resulting binary number by certain number. And then changing the final resulting binary number back to decimal number will give the shifted decimal number.

For example: 15<<3 (shift 15 by 3 to left). First change the decimal number 15 to its binary number 1111. This number has zeros as decimal points: 1111.00000. If we shift the decimal place by 3, we will get 1111000. If we change this number to decimal number, we will get 120. Therefore 15<<3= 120. Furthermore if you would want to compute 15>>3. The binary form of 15 which is 1111 will be changed to a binary number 1 and then to the decimal number1.

1.12.6. The right shift (>>) bitwise operator

To find the right shift of a decimal number by a certain value means to change the decimal number to binary and move back the decimal place of the resulting binary number by a certain number and changing the resulting number back to decimal. The resulting decimal number is the value we are looking for.

For example: 15>>3 (shift to right). First change 15 to binary: 1111. Then shift the decimal place by 3. The process will yield 1. Changing 1 that is in binary to decimal number will result in 1. Therefore 15>>3=1

Table 23: Bitwise operators

>>> ~14	>>> 12&14	>>> 12|14	>>> 15^10	>>> 15<<3	>>> 15>>3
-15	12	14	5	120	1
>>> ~41					
-42					

Python part 2

1. Math functions in python

To work with math functions in python we have to load math module by typing in >>> import math. When you are importing modules you can import them with alias. For example you can load math module in python with alias by typing in >>> import math as m. If you do not want to load the entire math module and only for example just the pow and sqrt function, restart your shell and type in >>> from math import sqrt,pow. If you want to study further about math functions type in >>> help('math') and investigate. You need to explore as many math functions as you can in help('math').

Table 24: Math functions

>>> import math	>>> import math as m	>>> from math import sqrt,pow
>>> x=math.sqrt(144)	#this is by using the concept of alias.	>>> pow(4,5)
>>> x	>>> math.sqrt(144)	1024.0
12.0	12.0	>>> pow(10,3)
>>> print(math.floor(2.1))	>>> m.sqrt(144)	1000.0
2	12.0	
>>> print(math.ceil(2.1))		
3		>>> help('math')
>>> print(math.pow(10,3))		Squeezed text (265lines.)
1000.0		(double click and find more
>>> print(math.pi)		information)
3.141592653589793		
>>> print(math.e)		
2.718281828459045		
>>>		

Students should explore further the help ('math') function of the IDLE and their math knowledge and study further about math functions in python.

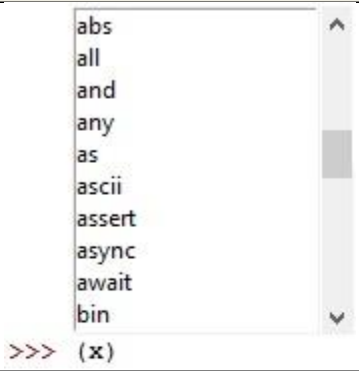
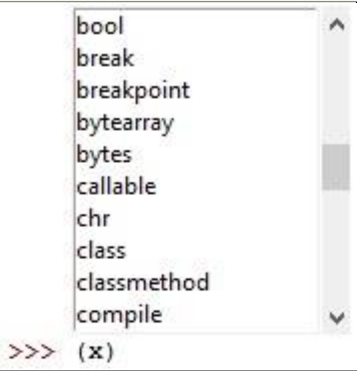
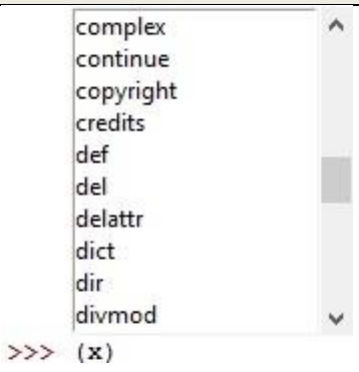
References

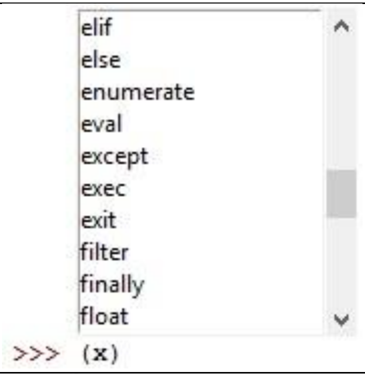
(Under development)

Appendix

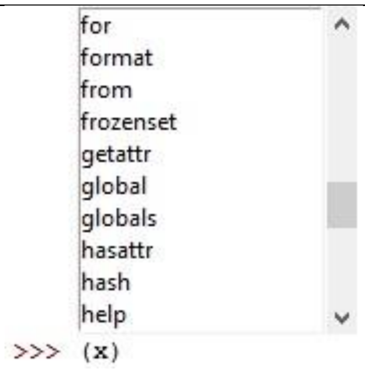
a. Methods

Table 25: Built-in methods

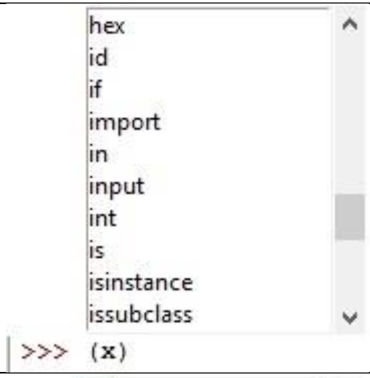
Function (value)		
 abs all and any as ascii assert async await bin >>> (x)	 bool break breakpoint bytearray bytes callable chr class classmethod compile >>> (x)	 complex continue copyright credits def del delattr dict dir divmod >>> (x)



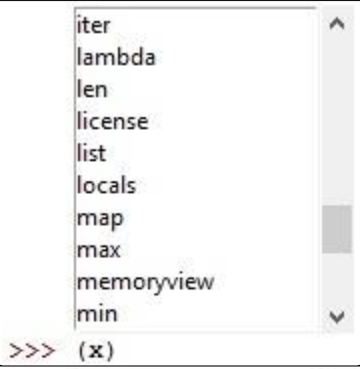
```
elif
else
enumerate
eval
except
exec
exit
filter
finally
float
>>> (x)
```



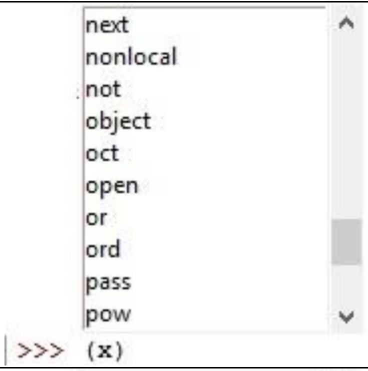
```
for
format
from
frozenset
getattr
global
globals
hasattr
hash
help
>>> (x)
```



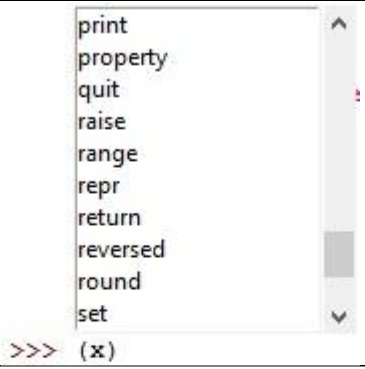
```
hex
id
if
import
in
input
int
is
isinstance
issubclass
>>> (x)
```



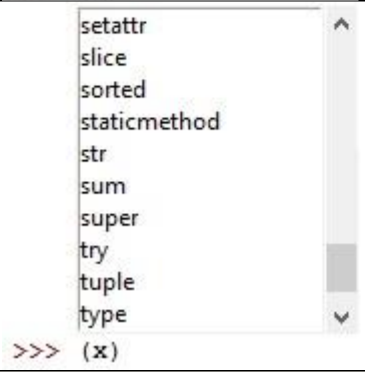
```
iter
lambda
len
license
list
locals
map
max
memoryview
min
>>> (x)
```



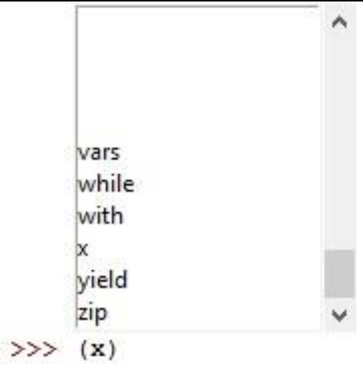
```
next
nonlocal
not
object
oct
open
or
ord
pass
pow
>>> (x)
```



```
print
property
quit
raise
range
repr
return
reversed
round
set
>>> (x)
```

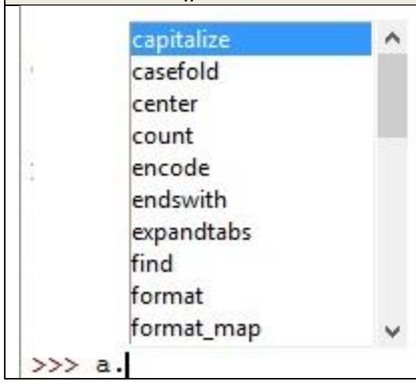


```
setattr
slice
sorted
staticmethod
str
sum
super
try
tuple
type
>>> (x)
```

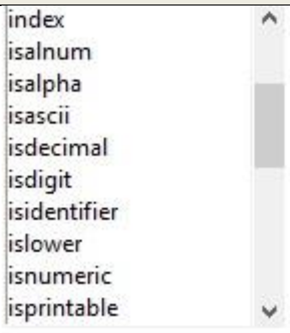


```
vars
while
with
x
yield
zip
>>> (x)
```

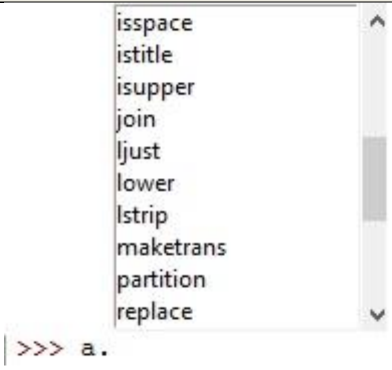
Vlue.function()



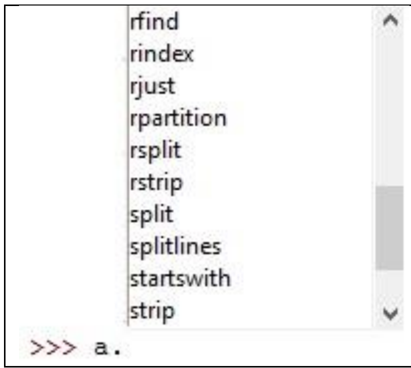
```
capitalize
casefold
center
count
encode
endswith
expandtabs
find
format
format_map
>>> a.
```



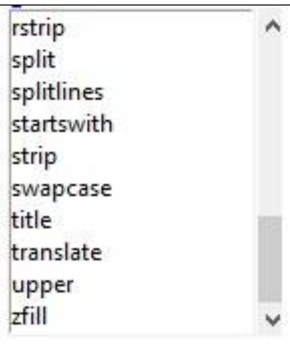
```
index
isalnum
isalpha
isascii
isdecimal
isdigit
isidentifier
islower
isnumeric
isprintable
```



```
isspace
istitle
isupper
join
ljust
lower
lstrip
maketrans
partition
replace
>>> a.
```



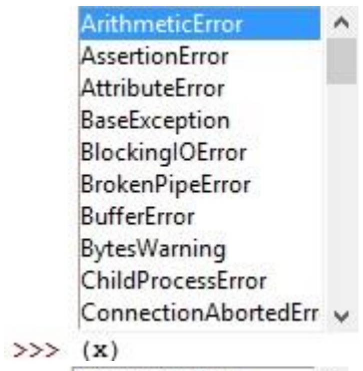
```
rfind
rindex
rjust
rpartition
rsplit
rstrip
split
splitlines
startswith
strip
>>> a.
```



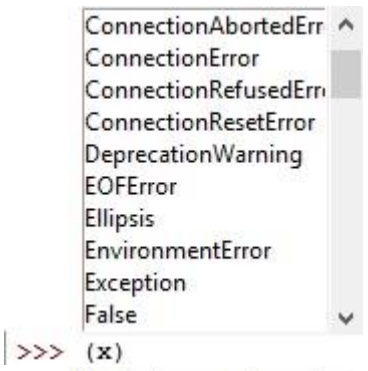
```
rstrip
split
splitlines
startswith
strip
swapcase
title
translate
upper
zfill
```

b. Errors

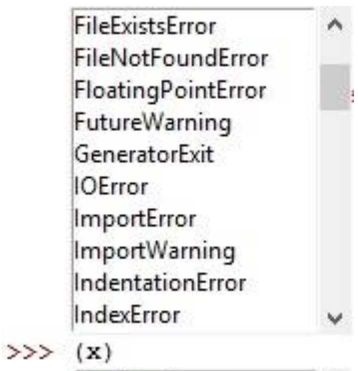
Table 26: List of errors



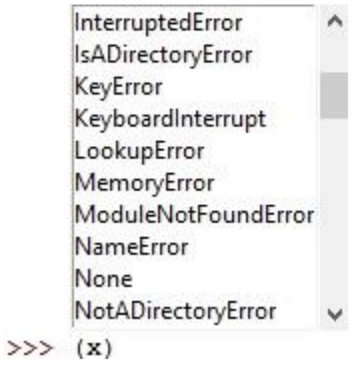
```
ArithmeticError
AssertionError
AttributeError
BaseException
BlockingIOError
BrokenPipeError
BufferError
BytesWarning
ChildProcessError
ConnectionAbortedErr
>>> (x)
```



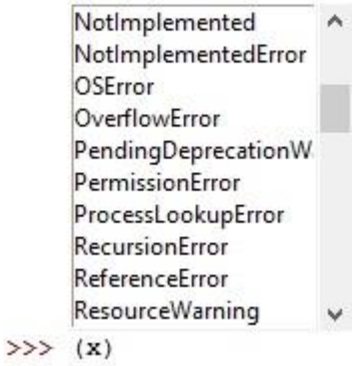
```
ConnectionAbortedErr
ConnectionError
ConnectionRefusedErr
ConnectionResetError
DeprecationWarning
EOFError
Ellipsis
EnvironmentError
Exception
False
>>> (x)
```



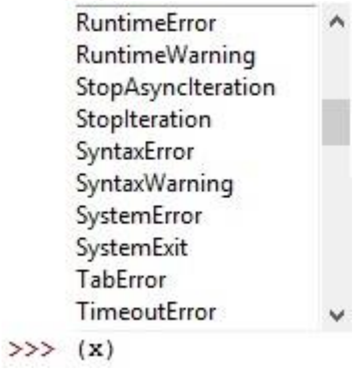
```
FileExistsError
FileNotFoundError
FloatingPointError
FutureWarning
GeneratorExit
IOError
ImportError
ImportWarning
IndentationError
IndexError
>>> (x)
```



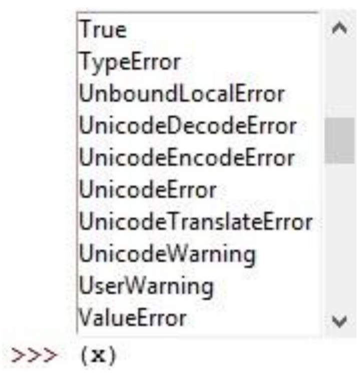
```
InterruptedError
IsADirectoryError
KeyError
KeyboardInterrupt
LookupError
MemoryError
ModuleNotFoundError
NameError
None
NotADirectoryError
>>> (x)
```



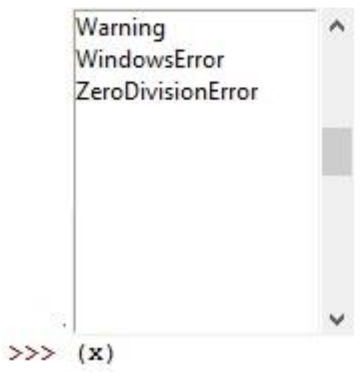
```
NotImplemented
NotImplementedError
OSError
OverflowError
PendingDeprecationW
PermissionError
ProcessLookupError
RecursionError
ReferenceError
ResourceWarning
>>> (x)
```



```
RuntimeError
RuntimeWarning
StopAsyncIteration
StopIteration
SyntaxError
SyntaxWarning
SystemError
SystemExit
TabError
TimeoutError
>>> (x)
```



```
True
TypeError
UnboundLocalError
UnicodeDecodeError
UnicodeEncodeError
UnicodeError
UnicodeTranslateError
UnicodeWarning
UserWarning
ValueError
>>> (x)
```



```
Warning
WindowsError
ZeroDivisionError
>>> (x)
```

c. Summary: part one and two

1. Recap (summary part 01):

Help:	***Operators in python:***	***Option + configure IDL:***	***Strings: (Immutable)***
help() TOPIC quit	(+ , -, *, /, //, %, **) There are operators for arithmetic, relational, logical and bitwise operations.	Change keys, etc. '_ ' (prints the previous output) UP key	'text' ; "text" ; 'text '*10 ; 'text' + ' text' ; \ ; 'it\'s all good!' \n : new line r : raw string syntax Sara's file is located in the computer's hard disk, C:\new-folder1\new-folder2.

….

The print function:	***Variables(1):***	***Variables(2):***	***Lists: (mutable)***

print('text, number, symbol')
print(operation)
	Assign values to strings, letters, and apply operators and other functions.
	Index:
	>>> t='eiabc'
	>>> t[4]>>'c'
	>>> print(t[4])>>c
	>>> t[0:3] >> 'eia'
	>>> t[0:]>>'eiabc'
	>>> t[:5]>>'eiabc'
	Length of a string:
	len(t)>5
	>>> t.count('i')>>1
	>>> t.replace('a','i')>>'eiibc'
	
	t + (.) + (ctrl+ spacebar) + select built in function + (text in quotations , operations and other functions)
		ID(TAG) and VALUE:
		>>> x=10
		>>> y=10
		>>>id(x)>>11711…
		>>>id(y)>>11711…
		Constant:
		>>> GRAVITY=9.8
		>>> type(GRAVITY)
		<class 'float'>
		>>> PI=3.14
		>>> type(PI)
		<class 'float'>
			>>> l=[7,9,8]
			>>> m=['a','c','b']
			>>> n=[1.0,'cat',8]
			>>> o=[2,0.0,3,1.5]
			>>> l+m+n>>[7, 9, 8, 'a', 'c', 'b', 1.0, 'cat', 8]
			>>> p=[l,m,n]
			>>> p>>[[7, 9, 8], ['a', 'c', 'b'], [1.0, 'cat', 8]]
			>>>
			>>> l.insert(0,'cat')
			>>> l>>['cat', 7, 9, 8]
			>>> m.sort()
			>>> m>>['a', 'b', 'c']
			>>> o.sort()
			>>> o>>[0.0, 1.5, 2, 3]
			>>> n.remove('cat')
			>>> n>>[1.0, 8]
			
			>>> l=['cat', 7, 9, 8]
			>>> l.remove('cat')
			>>> l>[7, 9, 8]
			>>> min(l)>>7
			>>> max(l)>>9
			>>> sum(l)>24
			>>> l.extend([10,11,12])
			>>> l>>[7, 9, 8, 10, 11, 12]
			>>>l.extend(['dog','cat','bird'])
			>>> l>>[7, 9, 8, 10, 11, 12, 'dog', 'cat', 'bird']

…

Tuple: immutable	***Set: mutable***	>>> s={1, 2, 3}

>>> t=(2,4,6,7)
>>> t>>(2, 4, 6, 7)
>>> t[3]>>7
>>> len(t)>>4
>>> t.index(6)>>2
>>> t.index(6,1,3)>>2
>>> t.index(6,0,1)
>>> t.count(4)>>1
	>>> s={1,2,3,4}
	>>> len(s)>>4
	>>> s.remove(4)
	>>> s>>{1, 2, 3}
	>>> r={3,4,5,6,7}
	>>> s.union(r)>>{1, 2, 3, 4, 5, 6, 7}
	s.intersection(r)>>{3}
		>>> r={3, 4, 5, 6, 7}
		>>> u={6, 7, 8, 9, 10}
		>>>s.intersection(r.intersection(u))
		set()
		>>> s.union(r.union(u))
		{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}

…

Data types:
 	
None
 	
Numeric= int, float, complex, bool
 	
Sequence= list, tuple, set, string, range
 	
Dictionary data types
	>>> x=10
	>>> type(x)>><class 'int'>
	>>> x=10.0
	>>> type(x)>><class 'float'>
	>>> x=3+8j
	>>> type(x)>><class 'complex'>
	>>> x=(2,5,8)
	>>> type(x)>><class 'tuple'>
	>>> x=[1,23,25,1]
	>>> type(x)>><class 'list'>
	>>> x={1,'cat',2.0}
	>>> type(x)>><class 'set'>
	>>> x='dog'
	>>> type(x)>><class 'str'>
		>>> x=10
		>>> y=2.3
		>>> z=int(x+y)
		>>> z>>12
		>>> z=float(x+y)
		>>> z>>12.3
		>>> x=10
		>>> y=2.5
		>>> z=x+y
		>>> x>y>>True
		>>> x<y>>False
		>>> x=y
		>>> x==y>>True
		>>> x>=y>>True
		>>> x<=y>>True

>>> x=range(4)
type(x)>><class 'range'>
>>> list(x)>>[0, 1, 2, 3,]
x=range(0,12,3)
type(x)>><class 'range'>
>>> list(x)>>[0, 3, 6, 9]
	x={'kebede':50/100,'Bekele':45/100,'Bele':65/100}
	>>> x.keys()>>dict_keys(['kebede', 'Bekele', 'Bele'])
	>>> x.values()>>dict_values([0.5, 0.45, 0.65])
	>>> x['kebede']>>0.5
	>>> x.get('kebede')>>0.5	

…

Assignment operators:
= , += , -= , *=, /=, %=
Relational operators:
< , > , <= , >=
Logical operators:
equality (== , !=) , logical AND (and), logical OR (or)
Postfix and Unary operators:
(++, --, !)
	>>>X=23
	>>> x+=2
	>>> x>25
	>>> x*=2
	>>> x>>50
	X=50
	>>> x-=2
	>>> x>>48
		>>> y=30
		>>> y/=3
		>>> y>>10.0
		>>> y=30
		>>> y%=3
		>>> y>>0
			>>>
			y,z=8,10
			>>> y>>8
			>>> z>10
			>>>
			x+z>>18
				>>> x=10
				>>> y=5
				>>> x>5>>True
				>>>
				x==y>>False
				>>> x>=y>>True
				>>>
				x<=y>>False
					>>> x=10
					>>> y=5
					>>> x>5 and
					y<6>>True
					>>>
					x==y>>False
					>>> x>20 or
					y>0>>True
					>>>
					x!=y>>True

…

Number system conversion; Binary (base 2(0,1)), decimal(base 10(0,1,2,3,4,5,6,7,8,9)), octal(base 8(0,1,2,3,4,5,6,7)), HexaDecimal (base 16(0,1,2,3,4,5,6,7,8,9,a,b,c,d,e,f))		

>>> x=12
>>> bin(x)
'0b1100'
	>>> hex(x)
	'0xc'
		>>> oct(x)
		'0o14'

….

Bitwise operators:
(Complement (~) (tilde operator) ; And(&) ; Or(|) ; XOR(^) ; Left Shift(<<) ; Right Shift(>>))
	>>> ~14
	-15
	>>> ~41
	-42
		>>>
		12&14
		12
			>>>
			12|14
			14
				>>>
				15^10
				5
					>>>
					15<<3
					120
						>>>
						15>>3
						0

2. Recap (summary part 02):

Math modules. First you can import math function by typing 'import math' in your IDLE. To access this information, type help('math') in your IDLE. You need to explore as many math functions as you can in help('math').

>>> import math
>>>math.sqrt(25)>>5.0
>>> math.floor(3.2)>>3
>>> math.ceil(3.2)>>4
>>> math.pow(10,2)>>100.0
>>> math.pi>>3.141592653589793
>>> math.e>>2.718281828459045
	>>> math.log10(100)>>2.0
	>>> math.log2(8)>>3.0
	>>>math.log(10,math.e)>>2.302585092994046
	>>> n=10
	>>> k=3
	>>> math.perm(n,k)>>720
