**Python Session 1 (20-Sep-20) (10AM to 1:10PM)**

Binary or Low Level

010111 1001

1010101

Advantages – Faster,

Disadvantages – Processor, OS

Middle Level or Assembly

8000 10

8001 20

8002 ?

Lxi HL, 8000

Mov B, M

Inc M

Mov A, M

Add B

Inc M

Mov M, A

+ve – Equally faster

-ve – Processor, OS

High – Level

Fortran, Pascal, Basic, Shell, C, C++, Java, C#, Python, Perl,

Extending, Embedding

Extensive Libraries

Numpy, scipy, matplotlib,

networking

Compiler – Translation of HL language code to machine level language (whole file)

Execution – Loading program to RAM

Interpreter – In built compilation and execution (line by line)

$ gcc helloworld.c

$ ./a.out

$ javac helloworld.java

$ java helloworld

|  |
| --- |
| $ cat helloworld.py  #!/usr/bin/python3  print ("Hello World! Welcome to Python!")  print ("Sum of 10 + 20 is ", 10 + 20)  $ chmod u+x helloworld.py  $ ./helloworld.py  $ #or  $ python3 helloworld.py |
| $ cat helloworld.c  #include<stdio.h>  int main()  {  printf("Hello World\n");  printf("Sum of 10 + 20 is %d\n", 10 + 20);  return 0;  }  $ gcc helloworld.c #Generates a.out if no errors  $ ./a.out |
| $ cat HelloWorld.java  public class HelloWorld {  public static void main(String[] args) {  System.out.println("Hello World\n");  return;  }  }  $ javac HelloWorld.java  $ java HelloWorld |

**Python**

1. Interactive Mode
2. Execution Mode
3. IDE (Integrated Development Environment)

IDLE

Identifiers or variable

1. Alphabet or numeric or \_
2. Start with alphabet or \_

Keywords

Data Types

1. Numbers (integers, floats, complex numbers)
2. Strings (“hello”, “python”, ‘world’, ”””My world”””)
3. Lists
4. Tuples
5. Dictionary
6. Sets
7. complex

Natural (1 to infinity)  
Whole (0 to infinity)

Integers (-infinity to 0 to infinity) – Radix 10

2 to 36 - Radix

Octal, Decimal, Hexa Decimal

12

Decimal (0 to 9)

Octal (0 to 7)

Hexa (0 to 9, A, B, C, D, E, F)

Conversions

Binary to Decimal (\* 2)

Decimal to Binary (/ 2)

Octal to Decimal (\* 8)

Decimal to Octal (/ 8)

Hexa Decimal to Decimal (\* 16)

Decimal to Hexa Decimal (/ 16)

**Python Session 2 - 26-Sep-20 (10AM to 1:10PM)**

Compiler – gcc – GNU C Compiler or GNU Compiler Collection

GNU – GNU Not Unix

Mutable – List, Dictionary, Set

Immutable – Integer, float, string, tuple

<https://towardsdatascience.com/https-towardsdatascience-com-python-basics-mutable-vs-immutable-objects-829a0cb1530a>

List

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 10 | 20 | 30 | 40 | 50 |
| 0 | 1 | 2 | 3 | 4 |
| -5 | -4 | -3 | -2 | -1 |

List []

Tuple ()

Dictionary, Set {}

L1[start:stop:step] from start to stop – 1

-1:-6:1 🡪 -1 < -6

-1:-6:-1 🡪 -1 > -6

Slicing is same as list for string and tuple

Dictionary 🡪 Key should be immutable (int, string)

Values 🡪 Integer, float, list, tuple, dictionary, set

Set 🡪 Collection of unique elements (unordered)

**Operators**

1. Boolean (True, False)
2. Arithmetic (+, -, \*, /, //, %, \*\*)
3. Assignment (Augmented Assignment) (=, +=, -= …) club with arithmetic or bitwise
4. Relational (<, >, <=, >=, ==, !=)
5. Logical (or, and, not)
6. Conditional ()
7. Bitwise (|, &, ^, ~, <<, >>)
8. Identity (id(), is, is not)
9. Membership (in, not in)

**Discussion as part of doubts clarification**

Library

Collection of functions

Math library

Modules (Collection of Functions) and Packages (Collection of modules)

Math is package, sqrt(), factorial(), pow(5, 3)

From math import sqrt, factorial, pow

**Python Session 3 – 27-Sep-20 (10:02 AM to 1:19PM)**

Augmented operator 🡪 Arithmetic, Bitwise

E1 or E2 🡪 If E1 is true, it won’t evaluate E2

E1 and E2 🡪 If E1 is false, it won’t evaluate E2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Bit1 | Bit2 | Bit1 | Bit2 | Bit1 & Bit2 | Bit1 ^ Bit2 | ~Bit1 |
| 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 1 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 0 | 0 |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 |
| 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 |
| 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |

V1 = 13 🡪 1101

V2= 11 🡪 1011

V1 | v2 🡪 15,

V1 & v2 🡪 9

V1 ^ v2 🡪 6

10 🡪 0000 1010

~10 🡪 1111 0101

-ve 🡪 2’s complement 🡪 1st complement + 1

0000 1010

+ 1

0000 1011

-11

-10 🡪 0000 1010 🡪 1111 0101 + 1 🡪 1111 0111

0000 1000 + 1 🡪 0000 1001 🡪 -9

1000 1011 🡪

0000 0000 1000 1011

1111 1111 0111 0100

0xFF76

0000 1100 🡪 12

0001 1000 🡪 24

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |
|  | 0 | 0 | 0 | 0 | 1 | 1 | 0 |

32 + 16 + 2

0011 0010

0001 1 001 ~~0 🡪~~25

12 🡪 shifting 3 left shift 12 \* ( 2 power 3)

N \* ( 2 power NSHIFTS)

100 🡪 shifting 3 right shift 100 / ( 2 power 3)

N / ( 2 power NSHIFTS)

Operator Precedence and association

**Conditional Statements**

if 🡪 if, if …. Else, if … elif … elif …. else

**Repetitive Statements or Iterative or Loops**

For in

While

Range -> start stop step

Range(10) 🡪 start = 0, step = 1

Range (2, 10) 🡪 step = 1

Range(3, 10, 2)

3

10

20

30

15, 23, 34, 45, 0

Loop Control Statements

S1, S2, S3 …. S10 🡪 Normal Way

S1, S2, S3, S4, S5, S6, S7, S8 🡪 Present, S9 and S10 are absent 🡪 break

S1, S2, S5, S6, S7, S9, S10 🡪 Present, S3, S4, S8 are absent 🡪 continue

Break 🡪 End of process

Continue 🡪 Ignore current iteration but proceed with next iteration

10 20 30 40 50 60 0

10 20 30 40 50 -60 0 🡪 Sum of +ve numbers

Code Generation 🡪 SymPy

**Python Session 4 – 4-Oct-20 (09:58AM to 12:45PM)**

Nested Loops

Loop within Loop

Any combination, Any levels

Sum of digits 🡪 1 + 2 + 3 + 4 + 5

5 + 4 + 3 + 2 + 1

**Functions**

Code …

Common Code …

Code ..

…

..

Common Code …

Common Code – 50 Lines, 20 Places (1000 Lines)

Function Definition:

Common Code

Function Call

Function/Processing Unit

Input Output

Optional Optional

Built-in Functions

User Defined Functions

1. No Input and No output
2. No input and Output
3. Input and No output
4. Input and Output

Fibo Series

1 1 2 3 5 8 13 21 34 55

N1 n2 sum

N1 n2 sum

Should not use below way:

def sum\_numbers4(n1=10, n2, n3=30):

  return n1 + n2 + n3

Default Arguments

Keyword Arguments

Variable Arguments

Variable Keyword Arguments

**Recursion**

Function calling itself

1. Base condition
2. Iterative Condition

5! = 5 \* 4!

4! = 4 \* 3!

..

1! = 1

|  |
| --- |
|  |
|  |
|  |
| Addr, num = 1 |
| Addr, num = 2 |
| Addr, num = 3 |
| Addr, num = 4 |
| Addr, num = 5 |

Call by value, Call by reference

Scope of a variable

**Python Session 5 – 10-Oct-20 (09:56AM to 1:03PM)**

Ascii Values

‘0’ – 48

‘9’ – 57

‘A’ – 65

‘Z’ – 90

‘a’ – 97

‘z’ – 122

Higher Order Functions

Comprehensions, Maps, Filter, Reduce

Object Oriented

Encapsulation, Inheritance, Polymorphism

Overloading, Constructor (\_\_init\_\_()) and Destructor (\_\_del\_\_()), add (\_\_add\_\_()), (\_\_len\_\_())

Files

Text, Binary

Open, read, write, close

W, r, w+, r+, a,

Seek

Fo.seek(3, 0)

0, 1, 2

0 – From beginning, 1 – from current location, 2 – from end

+ve – forward, -ve – backward