Python

Introduction:

- Python is low level language
- Python 3 is in use nowadays
- Python interpreter or Anaconda is used for python programming.
- Anaconda has libraries also.
- Python is an interpreter language.
- Implemented in C language.
- In Anaconda we use Jupyter Notebook.

★ Hello world program

```
print('Hello World')
```

output: Hello World

```
print('2+2 is' + str(2+2))
```

output:

2+2 is 4

★ Variable

- Used to make program generic like other languages.
- No need of declaring variables in Python.

Eg:

```
num1 = 4
num2 = 5
```

print (num1 + num2)

output: 9

- Also used for division and reminder operators.

Eg

```
print (num1 / num2)
print (num1 % num2)
```

- variables name can't be started with number.
- Can't be enclosed in quotation mark.
- Can't have any space in it.
- Can't be/begin-with number.
- Can't be any of Python's reserved words/keywords.

- Python is case sensitive language. (Num1 and num1 are different)

- '#' is used for comments in Python.

- Precedence of Python-arithmatic operators:

  * , / , // , % , + , -

. Compound assignment operators

num1 = num1 + 2
num1 += 2
} same

* Input

```
fname = input('First name: ')
lname = input('last name: ')
```

output:

```
fname : M H
lname : Gajoo
```

```
full_name = fname + lname
print('Full name is ', full_name)
```

output:

Full name is M H Gajoo

* Assignment operators

==
!=
>
<
=>
=<

Work as same as work in C or Java.

* If statement

```
if (num%.2 = 0):
        print("Number is even")
else
        print ("Number is odd")
```

- Works same as in other languages.
- (In this language if don't have brackets instead we have to skip blank space (tab) at the next line of if statement).

- Usually inputs in this language is string by default, so to make/convert it in integer we have to write following statement.

```
num = int(input('Enter a number'))
```

* Elif statement
- This statement works same as else-if statement in other languages.

* List

- In Python there are no arrays instead list is used.
- Unlike array list can be extended after initialization step.
- There are elements in ~~arr~~ list but necessisly of the same type like arrays.

~~Eg~~:

```
l = [1, 5, 7, 9]    #initialize list l
print(l[2])
```

output: 7

- Adding element in list

```
l.append(11)
```

(List became)

```
[1, 5, 7, 9, 11]
```

- Checking length of list

```
len(l)
```

output: 4

* Loop in list

```
for i in l:
    print(i)
```

output:

```
1
5
7
9
11
```

- For horizontal output:

```
for i in l:
    print (i, end=' ')
```

- Inserting element

```
l.insert (1,3)    # Insert 3 at 1st.
```

- To pop

```
l.pop()
```

★ Slicing in List

. Three parts

start, end, step

```
l = [1, 5, 7, 9, 11]    #List
l[1:3]                  # start, end
```

output: 5, 7

(Counts end number - 1)

```
l[1:]                   # start to list's end
```

output: 5, 7, 9, 11

```
l[0:5:2]                # start, end, step (gape of 2 elements
```

output: 1, 7, 11

- For reverse

  l[::-1]

- Same slicing can be applied over string.

  Eg:

  string = "The quick brown fox"

  string[5:10]

  output: 'uick'

* Tuple
  - Same as string, but it can't be modified (can't be appended, poped and so on).
  - Round brackets used for tuple.
  - East than List and so on.

* Dictionary
  - Stores data in the form of keys and values.

Eg:

d = {'noman': 80, 'ali': 30, 'MH': 89}

Here

Noman is key and 80 is value and so on.

- To check keys

```
d.keys()
```

output:

```
dict_keys(['noman' 'ali' 'MH'])
```

- To check value

```
d.value()
```

- To check value of a key

```
d['MH']
```

output 89

- Loop on dictionary

```
for k,v in d.items():
    print(k,' ',v)
```

* To convert types of input/value:

```
input()          # input
int()            #To convert in integer
float()          #          in float
str()                       in string
```

* Cases of strings

```
s.lower()
s.upper()
s.title()
```

* Function

- Making a function of repeatedly needed things and calling them where needed.

Eg:

```
def fun(num1, num2):
    return num1 + num2

fun(4, 6)          #invoked
```

output: 10

Eg:

```
def fun2(num1 = 8):        # setting default argument used when we don't give argument
    return num1 * num1

fun2(3)                    #invoked
```

output: 9

```
fun2()                     # invoked but didn't give argument
```

output: 64

- Unknown number of argument:

Eg:

```
def fun3(*nums):
    sum = 0
    for i in nums:
        sum += i
    return sum
```

fun3(3,4,7) #invoked #tuple

output: 14

(We can pass any number of argument)

Eg:

```
def fun4(**nums):
    for i in nums.keys():
        print(nums[i])

fun4(a=6, b=7, c=8)
```

output:
6
7
8

* While loop

```
a = 10
while (a<0):
    print(a)
    a = a-1
```

output:
10
9
:
1

* Range

Range also can be used in list.

Eg:

```
list(range(1,101)
```

output:
1
2
3
:
100

```
for i in range(1,101): #For horizontal and loop
    print(i, end = ' , ')
```

* OOp in python:
  - There is also class concept in python.

Eg:

```
class student:                  # Class
    def init_(self):            # class function
        self.id = 0
        self.name = 'roman'
    def enroll(self):
        print('studen', self.id, 'enrolled')

    def exam(self)
        print('student', self.id, 'is giving exam')

s = student(1, 'ali')           #object
print(s.id)
```

output: 1

```
print(s.name)
```

output: Ali

```
s.enroll()                      # call
```

output: student is enrolled

* Sub class:

```
class ResearchStudent (student):
    def __init__(self, id, name, research_title):
        super(), __init__(id, name)
        self.research_title = research.title

r=ResearchStudent (5, 'asgher', 'Network')  #making obj
r.research_title
```

output: Networks

```
r.id
```

output: 5

```
r.name
```

output: Asgher

Python (Classwork)

=> Python Intro and descriptions avalaible in slides.

* Program steps or program flow
  - Sequnted structure
  - Selection/conditional structure.
  - Repeatation structure

All of these are described in slides.

* Reserved word

In python there are many reserved words that we can't use as variable names / identifiers.

=> All are mentioned in slides.

* Constants

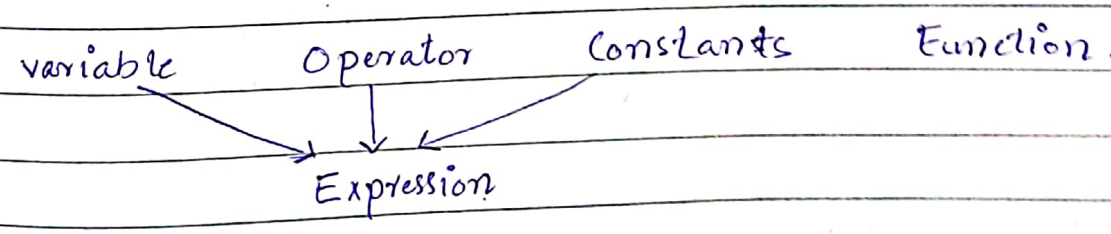
mentioned in slides.

* Variable

In slides.

* Sentences or lines

x = 2	Assignment statement
x = x+2	Assignment with expression
print(x)	Print statement



* Programming paragraphs

A group of statements to express any complicated form.

* Python scripts
  Expressed in slides.

* Interactive vs scripts.

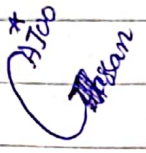
- Interactive
  - You type directly to python one line at a time and it responds.
    (Avalaible in latest version of java.
- Scripts
  You enter a sequence of statements into a file using a text editor and tell python to execute the statements in the file

* Python variable name rules
  - Slides

* Mnemonic variable names
  - slide
  - Good variable names tells us what are these bits of code doing.

* Assignment statements
  - Slides

* Expressions
  - Slides



* Numeric Expression
  - Slides
  - In division it is dejault that it always gives result in float.

* Order of evaluation
  - Slides

* Operator precedence rules
  - slides

Paranthesis
Power
Multiplication / Division
Addition / substraction
(Left to right)

* Type

In Python variables, Literals and constants have a type.

- More in slide

=> Concatenate means put together.

* Type Matters
  - Slide

type (variable)
output : type of variable.

* Several types of numbers
  Num - Two main types
  Integer
  Floating point

* Type conversion
  - Slide

=> Less accurate types are converted automatically in more accurate types when used together.
That's called implicitly conversion.

=> Built in functions int(), float()

* String conversion
  - Slide

* User input
  - Input ()
  - Slides.

* Converting user input
  - Slides

* Comments in python
  - Single line start with #.
  - for multiline """cmnt"""

* Conditional Execution

* Conditional Steps
  - Slide

- Some rules are

-> Use colon ':' in the end of if or anyother block starting. :

-> Skip some space (indent) on next line.

-> If there is only one statement then we can write on same line.

* Comparison Operators
  - Slide

=> In conditional statements there are
  -> One way decisions
  -> Two way decisions
  -> Mult way decisions

=> In one way decision there is only one option / Block to execute and there is not any other option like 'else' etc.

=> In two way decision there is another option 'else' after if statement, if the block 'if' is not true the block 'else' executes.

=> In multi way decision there are more than two options like there is 'elif' statement after 'if' and there may be more than one 'elif' statements and in last we can use 'else' statement.

* Indentation
  - Slide

=> If the block/statement 'if' is true the interpreter will ignore all remaining blocks related to that 'if' statements.

=> If is essential in one way decision.
=> If and else statements are essential in two-way.
=> If and elif statements are essential in multiway decision. (else not essential).

* Nested decisions:
  - Slide

* The Try / Except Structure:
  - When we expect runtime error and we want that the execution of the program should not be inhibited/stoped then we use try/except structure.
  - If the code in try works - the except is skipped.
  - If the code in try fails - It jumps to the except section.

* => Exercise