```
print("Hello, World")
```

```
Hello, World
```

```
# int float str
# int- integer
# float - decimal - 0.1,0.01,-0.1,10.3
# str - character - "a",'a',"bus","im in a bus"," ","#","@"
# ctrl+ A (select all)
# ctrl+ / (comment all)
```

```
# Variables
```

```
100+25+40
```

```
165
```

```
100*7
```

```
700
```

```
100%7
```

```
2
```

```
virat = 60
rohit = 30
rahul = 10
```

```
print(virat+rohit+rahul)
```

```
100
```

```
x=1
y=2
z=3
print(x,y,z)
```

```
1 2 3
```

```
# multiple variables assign values once
x,y,z=1,2,3
print(x,y,z)
```

```
1 2 3
```

```
a,b=10,20
c=a+b
c
```

```
30
```

Syntax refers to set of rules that defines the combinations of symbols that are considered to be correctly structured programs in a langugae.In simple term syntax is about the correct arrangement of words and symbols in a code.

Sementaics refers to the meaning or interpretation of symbols,characters and commands in a language.

```
# BAsic Syntax Rules in Python
# case sensitivity - Python is case sensitive
name="Siva"
Name="Varma"
print(name)
print(Name)
```

```
Siva
Varma
```

Indentation in python is used to define structure and hierarchy of the code.Unlike many other programming languages that use braces {} to delimit blocks of code.

```
# Indentation
# Python uses indentation to define blocks of code. Consistent use of spaces (commonly 4) or a tabs i re
age =32
if age>30:
  print(age)
print(age)
```

```
32
32
```

```
# this is a single line comment
print("Hello ,  World")
```

```
# this is a multiline comment
'''
this is an example of multiline comment
'''
```

```
## Line Continuation
total=1+2+3+4+5+6+7+ \
4+5+6
print(total)
```

```
43
```

```
## Multiple statements on a single line
x=5;y=10;z=x+y
print(z)
```

```
15
```

```
## understand Semantics in python
# variable assignment
age=32 ## age is an integer
name="SIva" ## name is a string
print(age)
print(name)
```

```
32
SIva
```

```
# type()- function used to know data type of each variable
print(type(age))
print(type(name))
```

```
<class 'int'>
<class 'str'>
```

```
age=32
if age>30:
print(age)# indentation is important
```

```
  File "<ipython-input-8-31f9d94f29be>", line 3
    print(age)
    ^
IndentationError: expected an indented block after 'if' statement on line 2
```

Next steps: Explain error

```
# Name error
a=b # b is not defined any value or assigned as variable
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-9-607beccfde8a> in <cell line: 0>()
      1 # Name error
----> 2 a=b

NameError: name 'b' is not defined
```

Next steps: Explain error

```
# code examples of indentation
if True:
  print("Correct Indentation")
  if False:
    print("This onto print")
  print("This will print")
print("Outside the if block")
```

```
Correct Indentation
This will print
```

```
Outside the if block
```

Variables

```
# Variables are fundamental elements in programming used to store data that can be
# referenced and manipulated in a program.In Python variables are created when
# you assigna a value to them,and they do need explict declaration to reserve memeory space
```

Introduction to variabels

Decalring and assigning variables

Naming Conventions

Understanding Variable Type

Dynamic Typing

```
a=100
```

```
# Decalring and assigning Variables
```

```
age=32
height=6.1
name="Siva"
is_student=True
print("age:",age)
print("height:",height)
print("Name:",name)
```

```
age: 32
height: 6.1
Name: Siva
```

```
## Naming Conventions
## VAriable names should be descriptive
## THey must start with a letter or an '_' and contains letter,numbers and underscores
## Variables names case sensitive
#valid variable names
first_name="Siva"
last_name="Varma"
```

Double-click (or enter) to edit

```
# invalid way of writing a variable by starting with a number
# 2age=30
```

```
#first-name="Siva"
# @name="SIva"
```

```
  File "<ipython-input-13-b47b58950427>", line 2
    2age=30
    ^
SyntaxError: invalid decimal literal
```

Next steps: Fix error

```
# Python is dynmaically typed,variable is defined at runtime
age=25#int
height=6.1#float
name="Siva"#str
is_student=True#bool
print(type(is_student))
```

```
<class 'bool'>
```

```
# Type conversion
age=25
print(type(age))
# Type conversion of int to str
age_str=str(age)
print(type(age_str))
```

```
<class 'int'>
<class 'str'>
25
```

```
name="Siva"
int(name)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-19-22f6eee05d34> in <cell line: 0>()
      1 name="Siva"
----> 2 int(name)

ValueError: invalid literal for int() with base 10: 'Siva'
```

Next steps: Explain error

we cannot convert str to int if it is alphabetical

```
height=5.11
print(type(height))
```

```
<class 'float'>
```

```
print(int(height))
print(type(int(height)))
```

```
5
<class 'int'>
```

```
## Dynamic typing
var=10#int
print(var,type(var))
```

```
10 <class 'int'>
```

```
var=3.14
print(var,type(var))
```

```
3.14 <class 'float'>
```

```
age=input("WHat is the age")
print(age,type(age))
```

```
WHat is the age23
23 <class 'str'>
```

```
### SImple calculator
num1=float(input("Enter the first number: "))
num2=float(input("Enter the second number: "))
sum=num1+num2
difference=num1-num2
product=num1*num2
quotient=num1/num2
print("Sum:",sum)
print("Difference:",difference)
print("Product:",product)
print("Quotient:",quotient)
```

```
Enter the first number: 12
Enter the second number: 12
Sum: 24.0
Difference: 0.0
Product: 144.0
Quotient: 1.0
```

DATA TYPES

1.Defintion

Data types are classification of data which tell the compiler or interpreter how the programmer intends to use the data.

2.Importances of data types in programming

Data types ensure that data is stored in an efficient way.They help in performing correct operation on data.

```
# Integer data type
age=23
type(age)
```

```
int
```

```
# floating data type
height=5.11
print(height)
print(type(height))
```

```
5.11
<class 'float'>
```

```
# string data type example
name="Siva"
print(name)
print(type(name))
```

```
Siva
<class 'str'>
```

```
# boolean datatype
is_true=True
print(is_true)
print(type(is_true))
```

```
True
<class 'bool'>
```

```
a=10
b=10
print(a==b)
```

```
True
```

```
# common error
result="Hello"+5
result
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-33-7d1c6f4b8014> in <cell line: 0>()
      1 # common error
----> 2 result="Hello"+5
      3 result

TypeError: can only concatenate str (not "int") to str
```

Next steps: Explain error

```
result="Hello"+str(5)
result
```

```
'Hello5'
```

Deep Dive into operators

1.Introduction to operators

2.Arthimetic Operators

Addition,Subtaction,Multiplication,Division,FloorDivision,Modulos,Exponentiation

3.Comparison Operators

Equal to,Not equal to,Greater than or equal to ,less than or equal to

4Logical operator

AND,OR,NOT

```
## Arthemitc Operators
a=10
b=5
add_result=a+b
sub_result=a-b
mul_result=a*b
divi_result=a/b# float type as output
floor_div_result=a//b# int type as output
mod_result=a%b
expo_result=a**b
print(add_result)
print(sub_result)
print(mul_result)
print(divi_result)
print(floor_div_result)
print(mod_result)
print(expo_result)
```

```
15
5
50
2.0
2
0
100000
```

Comparison operator

```
##==
a=10
b=10
```

```
print(a==b)
```

```
True
```

```
str1="Siva"
str2="Siva"
print(str1==str2)
```

```
True
```

```
# Not Equal to !=
str1!=str2
```

```
False
```

```
str3="Siva"
str4="Siva"
str3!=str4# both of them are equal so False
```

```
False
```

```
str3="Siva"
str4="siva"
str3!=str4
```

```
True
```

```
# Greater than >
num1=45
num2=35
num1>num2
```

```
True
```

```
# less than <
num1=45
num2=35
num1<num2
```

```
False
```

```
# greater than or equal to
num1=45
num2=45
num1>=num2
```

```
True
```

```
num1=45
num2=45
num1<=num2
```

```
True
```

Logical operator

```
#And ,Not,Or
# AND - if a and b are True then only Ture
# OR  - if one condtion is True it is True
```

```
x=True
y=True
result=x and y
print(result)
```

```
True
```

```
x=True
y=False
result=x and y
print(result)
```

```
False
```

```
x=True
y=False
result=x or y
print(result)
```

```
True
```

```
# Not operator opposite
x=True
not x
```

```
False
```