Basic Python

Syntax and Semantics in Python

```
## Basic Syntax Rules In Python
## Case sensitivity- Python is case sensitive

name="Kiran"
Name="Ram"

print(name)
print(Name)
```

```
Kiran
Ram
```

Indentation

Indentation in Python is used to define the structure and hierarchy of the code. Unlike many other programming languages that use braces {} to delimit blocks of code, Python uses indentation to determine the grouping of statements. This means that all the statements within a block must be indented at the same level.

```
## Indentation
## Python uses indentation to define blocks of code. Consistent use of spaces (commonly 4) or a tab is required.

age=32
if age>30:

    print(age)

print(age)
```

```
32
32
```

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

```
Hello World
```

```
'''
This is multi-line comment
This is multi-line comment
This is multi-line comment
This is multi-line comment
'''
print("Hello World")
```

```
Hello World
```

```
## Line Continuation
##Use a backslash (\) to continue a statement to the next line

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  Semnatics In Python
# variable assignment
age=32 ##age is an integer
name="Krish" ##name is a string
```

```
# Check data-type of variable
type(age)
```

```
int
```

```
type(name)
```

```
str
```

```
## Type Inference / Dynamic typing
variable=10
print(type(variable))
variable="Krish"
print(type(variable))
```

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

```
## Name Error
a = b
```

```
----------------------------------------------------------------------
-----
NameError                                 Traceback (most recent call
last)
```

```
Cell In[11], line 2
      1 ## Name Error
----> 2 a = b

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

```python
## Code exmaples of indentation
if True:
    print("Correct Indentation")
    if False:
        print("This ont 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 assign a value to them, and they do not need explicit declaration to reserve memory space. The declaration happens automatically when you assign a value to a variable.

```python
a = 100

## Declaring And Assigning Variables

age = 32
height = 6.1
name = "Kamal"
is_student = True

## printing the variables

print("age :",age)
print("Height:",height)
print("Name:",name)
print("is_student:",is_student)
```

```
age : 32
Height: 6.1
Name: Kamal
is_student: True
```

```python
## 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 = "Ramesh"
last_name = "Kadam"

# Invalid variable names
# 2age = 30
# first-name = "Karim"
@name = "Kajal"
```

```
  Cell In[17], line 4
    @name = "Kajal"
     ^
SyntaxError: invalid syntax. Maybe you meant '==' or ':=' instead of '='?
```

```python
## case sensitivity
name = "Neelima"
Name = "Nayan"

## Understnading Variable types
## Python is dynamically typed,type of a variable is determined at runtime

age = 25 #int
height = 6.1 #float
name = "Samir" #str
is_student = True #bool

## Type Checking and Conversion

age=25
print(type(age))

# Type conversion
age_str=str(age)
print(age_str)
print(type(age_str))
```

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

```python
age='25'
print(type(int(age)))
```

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

```
name = "Lalita"
int(name)
```

```
---------------------------------------------------------------------------
-----
ValueError                                Traceback (most recent call
last)
Cell In[22], line 2
      1 name = "Lalita"
----> 2 int(name)

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

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

```
float
```

```
float(int(height))
```

```
5.0
```

```
## Dynamic Typing
## Python allows the type of a vraible to change as the program
executes
var=10 #int
print(var,type(var))

var="Hello"
print(var,type(var))

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

```
10 <class 'int'>
Hello <class 'str'>
3.14 <class 'float'>
```

```
## input
```

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

```
What is the age20
20 <class 'int'>
```

```
### Simple calculator
num1 = float(input("Enter first number: "))
num2 = float(input("Enter 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 first number: 45
Enter second number: 2
Sum: 47.0
Difference: 43.0
Product: 90.0
Quotient: 22.5
```

DataTypes

1. Definition:

Data types are a classification of data which tell the compiler or interpreter how the programmer intends to use the data. They determine the type of operations that can be performed on the data, the values that the data can take, and the amount of memory needed to store the data.

1. Importance of Data Types in Programming Explanation:

- Data types ensure that data is stored in an efficient way.
- They help in performing correct operations on data.
- Proper use of data types can prevent errors and bugs in the program.

Introduction to Data Types

- Basic Data Types
  - Integers
  - Floating-point numbers
  - Strings
  - Booleans
- Advanced Data Types
  - Lists
  - Tuples
  - Sets
  - Dictionaries
- Type Conversion

```
## Integer Example
age=35
type(age)
```

```
int
```

```
##floating point datatype
height=5.11
print(height)
print(type(height))
```

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

```
## string datatype example
name="John"
print(name)
print(type(name))
```

```
John
<class 'str'>
```

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

```
bool
```

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

```
type(a==b)
```

```
bool
```

```
## common errors

result = "Hello" + 5
```

```
----------------------------------------------------------------------
-----
TypeError                                 Traceback (most recent call
last)
Cell In[33], line 3
      1 ## common errors
----> 3 result = "Hello" + 5

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

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

```
Hello5
```

Operators

1. Introduction to Operators

2. Arithmetic Operators

- Addition
- Subtraction
- Multiplication
- Division
- Floor Division
- Modulus
- Exponentiation

1. Comparison Operators

- Equal to
- Not equal to
- Greater than
- Less than
- Greater than or equal to
- Less than or equal to

1. Logical Operators

- AND
- OR
- NOT

1. Practical Examples and Common Errors

```
## Arithmethic Operation

a=10
b = 5

add_result=a+b  #addiiton
sub_result=a-b  #substraction
mult_result=a*b #multiplication
div_result=a/b  #division
floor_div_result=a//b ## floor division
modulus_result=a%b #modulus operation
exponent_result=a**b ## Exponentiation

print(add_result)
print(sub_result)
print(mult_result)
print(div_result)
print(floor_div_result)
print(modulus_result)
print(exponent_result)
```

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

```python
# Comparison Operators
```

```python
## Comparison Operators
## == Equal to
a=10
b=10
```

```python
a==b
```

```
True
```

```python
str1="Kira"
str2="Kira"
```

```python
str1 == str2
```

```
True
```

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

```
False
```

```python
str3="John"
str4="john"
```

```python
str3 != str4
```

```
True
```

```python
# greater than >
```

```python
num1=45
num2=55
```

```python
num1>num2
```

```
False
```

```python
## less than <
print(num1<num2)
```

```
True
```

```
#greater than or equal to
number1=45
number2=45

print(number1>=number2)

True

#less than or equal to
number1=44
number2=45

print(number1<=number2)

True

# Logical Operators

## And ,Not,OR
X=True
Y=True

result =X and Y
print(result)

True

X=False
Y=True

result =X and Y
print(result)

False

## OR
X=False
Y=False

result =X or Y
print(result)

False

# Not operator
X=False
not X

True

## Simple Calculator

# Simple calculator
```

```python
num1 = float(input("Enter first number: "))
num2 = float(input("Enter second number: "))

# Performing arithmetic operations
addition = num1 + num2
subtraction = num1 - num2
multiplication = num1 * num2
division = num1 / num2
floor_division = num1 // num2
modulus = num1 % num2
exponentiation = num1 ** num2

# Displaying results
print("Addition:", addition)
print("Subtraction:", subtraction)
print("Multiplication:", multiplication)
print("Division:", division)
print("Floor Division:", floor_division)
print("Modulus:", modulus)
print("Exponentiation:", exponentiation)
```

```
Enter first number: 16
Enter second number: 3
Addition: 19.0
Subtraction: 13.0
Multiplication: 48.0
Division: 5.333333333333333
Floor Division: 5.0
Modulus: 1.0
Exponentiation: 4096.0
```