CS 327: Assignment 1
Topic: CodeSangam Language Syntax

Instructor: Prof. Abhishek Bichhawat

Group: Comparsers

Name	**Roll number**
Dhruv Gupta	21110070
Hirva Patel	21110154
Shubh Singhal	21110206
Disha Chopra	21110057

Definition of Different Data Types

Variable names must start from an alphabet and can contain only alphanumeric characters and underscore (_).

We are hosting three different datatypes:

Number (int and float): `num`

Boolean: `bool`

String: `str`

Declaration statements:

```
num a = 99
str c = "exp"
bool x = Sahi/Galat
var v = "string"  (to assign datatype based on expression) (to include " in str use \")
var v = 23        (num/bool)
```

Redeclaration of variables is not allowed, and a variable, once assigned to one data type, can't be assigned to a different datatype.

Compound Structures

```
dict<num,num> d = [2:3, 3:4, 5:6, 7:8 ]          (keys can num and str only)
dict<num,dict<num,num>> d2 = [2:[3:4,4:5], 0:[1:2, 2:3]]
```

Dictionary Functions:

`d[key] = "Camel"`	(if str)
`d[key] = 2346`	(if num/bool)
`d.keys()`	returns a list of keys
`d.pop(key)`	remove the given key-value pair
`d.val()`	returns a list of values
`dict d3 = d.copy()`	makes a copy of dictionary
`d.len()`	returns the number of keys

```
list<str> l = ["Hirva", "Disha", "Dhruv", "Shubh"]
```

List functions:

`l.append(123)`	if num/bool/var
`l.append("str")`	if str
`l.insert([0],24)`	inserts 24 at 1st position
`l.join(l2)`	adds l2 after of l
`num x = sum(l)`	returns sum if list datatype is num
`l.len()`	returns the size of l

`l.count(1)`	counts the occurrences of 1
`l.index(4)`	returns element at 5th position
`l.slice[1:5]`	takes 2nd, 3rd, 4th and 5th element
`l.index(34)`	returns the first occurrence index of 34

`tuple<var> t = ["abc",Galat,69]` (tuple<var> allows different datatypes in its content)

Tuple Functions:

`t.count(1)`	counts occurrences of 1
`t.index(43)`	returns index of 43
`t.len()`	returns the length of the tuple
`t(3)`	accesses 4th element

Operations

Number operations:

`num x`	garbage values
`y = int(x)`	to concatenate a float to int, but the datatype will be num
`str c = a+b`	
`x = y+z`	
`x += y`	
`x = y//z`	integer divide
`x = y%z`	returns remainder

.

String operations:

`str c = a+b`	concatenate
`str c = a[1:5]`	slicing

Conditional Operators: Used for comparison between 2 operands/ expressions.

<	: less than
>	: greater than
>=	: greater than or equal
<=	: less than or equal to
==	: is equal?
!=	: not equal to

Print Statement

It is the keyword print itself, with all functionalities like Python, where we don’t have to specify the variable's data type to be printed explicitly.

Prints in new line every time.

```
str name = Dhruv
print(“Hello ”,name)                                 >>Hello Dhruv
print(“hello”,end=””)                                >> ends with “”
```

If Statement

Keyword for if is agar, elif is magar and else is nahitoh. Each conditional is followed by a colon. Further, no brackets are required instead only indentation works.

```
    agar (marks>80):
        print(“Pass”)
    magar (marks>30 && marks<=80):
        print(“Re-exam”)
    nahitoh :
        print(“Fail”)
```

An alternative for the break keyword is “niklo”.

Loops: For and While

The keyword for “for” is “keliye” and “ while” is “jabtak”. No brackets are required instead indentation has to be followed.

```
    num count=0
    keliye (num i=1; i<9;  i++):
        statements
    nahitoh:                (code enters this when loop exits normally)
        statements

    jabtak ((condition) == Sahi):
        grade++
    nahitoh:
        Statements
```

Function definition

The keyword for defining a function is “karya”. Also, function declaration is done by keyword followed by the function name, brackets and colon.

```
karya complierProject(bool: works):
     num grade
     agar (works==Sahi):
          grade = 11
     nhitoh:
          grade = 9
     vapas grade
```

Closures

A closure is a function object that has access to variables in its lexical scope, even when the function is called outside that scope.

```
karya outerKarya():
     num outVar = 1
     karya innerKarya():
          print(outVar)
     vapas innerKarya
```

Mutable variables

Lists and dictionaries are mutable type objects.

Let Statements

We can use variables inside print statements using let functions.

```
print((let a = 5 in a) * (let a = 6 in 2*a))
```

This statement will print 60.

Exceptions

Similar to try-except for python

```
koshish:
    compilerProject()
warna:
    gradedown()
```