Hinglish WASM

Statements in our language end with End of Line. Indentations are used to identify the scope of a statement. Indentations can be four spaces or tabs.

Definitions of different data types:

Variable names have to start from an alphabet and can contain only alphanumeric characters.
We are hosting 3 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.

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[2]            indexing (0-based indexing)
str c = a[1:5]          slicing
```

Compound Structures (dictionaries, lists, tuples, arrays)

```
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
d2.join(d)	adds d to d2

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 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 first occurrence index of 34
l.remove(34)	removes 34 in list
l.sort()	sorts with ASCII values, only when num or str

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

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

Conditionals:

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

An operation that prints values

is the keyword print itself, with all functionalities like Python where we don’t have to specify the data type of the variable to be printed explicitly. (ends with EndOfLine)
Prints in new line everytime.
str name = Dhruv

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

If statements:

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”)
```

alternative for break keyword is niklo

Loops: for and while

Keyword for “for” is keliye and “ while” it 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:

Keyword for defining a function is karya. Also function declaration is done by keyword followed by 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, dictionaries are mutable type objects

Let Statements

print((let a = 5 in a) * (let a = 6 in 2*a)) (shall return 60)

Exceptions

Similar to try-except of python

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