Hydra

# **Description and Extension**

Hydra is a python-format general-purpose programming language that has the .hd extension in its source file. It is inspired by python, which is short, simple, and easy to read, however, hydra makes the simplicity of python, even more simpler and easier to read with a dynamically typed structure and only three simple data types (float, character, and bit). Hydra also enhances its execution efficiency, unlike python which uses an interpreter the execute line by line hydra takes the source code and executes it all in one go. This language takes the Boa source code and converts it to python programming language format which is more human-readable and user-friendly.

# **Name/Extension**

The name of the language is **Hydra** which is named after an immortal, many-headed snake whose heads grew again as they were cut off, eventually killed by Hercules. Extension of the language is **.hd** in its source code. Hydra is a python-format general-purpose programming language and it has python-like functionalities in more simpler way.

# **Comments**

Hydra follows python-format for commenting on the source code. In order for the user to put a comment on something, they can simply start with putting the # followed by the comment they want to put in the source code file. Unlike python hydra support multi-line comments in order to do multi-line comment we start /# and close with #/.

# **Keywords**

The lists of hydra keywords are as follows:

* **print**  (output the result into the console for the user)
* **input** (takes input from the keyboard)
* **or** (logical keyword or get true only if one of the conditions is true)
* **and** (logical keyword and all the conditions should be true to become true)
* **not** (logical keyword not, inverse the condition)
* **if** (the if keyword is used in control statements to match )
* **elif** (else if keyword for the control statement)
* **else** (else control statement keyword if none of the conditions match)
* **for** (for control structure keyword to keep iterating till its range or its condition)
* **while** (while control structure keyword, keep iterating till the condition becomes false)
* **is** (to match if one variable or value is equal to another)
* **false** (logical data type false )
* **true** (logical data type true )
* **break** (break get out a loop or any other control structure)
* **return** (return keyword end the function return the value of a function)
* **def** (def to define a function )

# **Variables and Data Types**

Hydra has only three primitive data types. These data types are numeric, character, and bit (true/false).

The numeric data types are double precision which is eight bytes and can be used for both integers and floating-point numbers. Character datatype can be both single characters or string of characters. Bit character holds only 0 or 1 value in it.

E.g.

A = “Hasib”

PI = 3.134

result = false

# **Assigning values**

To assign a value to the variable the assignment operator (=) is required to assign a value to the variables. Users can assign constant literals by assignment operator; all the data types are dynamic based on the value that they are assigned data types take that form.

E.g.

A = “Hasib”

PI = 3.134

result = false

# **Selection Statements**

The selection structure is the same as python using the keyword if, elif, and else the program can make a decision between different options and select.

Eg:

X = 15

Y= 12

If x == y:

print(“x is equal to y”)

elif x > y:

print(“X is greater than y”)

else:

print(“X is less than y”)

# **Iteration**

Iteration in hydra is the same as iteration in python using *for loop* and *while loop.* In order to use the for loop the user first needs a condition to be true so that while loop keeps iterating until the condition becomes false.

E.g.

I = 4

While I > 1:

print(i)

i—

For loop, the user can specify a range to keep iteration until that range

E.g.:

* For i in range(10):

print(i)

* for i in range(10,20)

print(i)

* for i in range (10,30,4)

print(i)

# **Function**

To define a function, we use the def keyword followed by the function name and the return statement is dynamic. If the user does not include the return statement, the function will be void.

E.g.

* def func\_name():

print(“Hello world”)

* def func\_Name(a,b):

return (a+b)

**GITHUB:**

**https://github.com/Khesorw/Compilers.git**