REPL

Read Eval Print Loop

## Data Types:

automated (decided by the value)

Numbers:

int integers

whole numbers

float floating point numerals

decimal points

complex

imaginary

7 + 6i

strings:

text data

## Features:

multiple assignment

## general functions:

print

type

len

sorted

all

any

max

min

bin

input

## Operators:

### arithmetic:

+ add

- subtract

\* multiply

/ division

= assigns

% modulo (remainder)

// floor division

\*\* exponent

### relational operators (comparative):

< lesser than

> greater than

<= lesser than or equal to

>= greater than or equal to

== equal to

!= not equal to

### logical:

and

or

not

### bitwise operator:

& and

| or

^ ex-or

>> right shift

<< left shift

## 

## Strings:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| b | a | n | g | a | l | o | r | e |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| -9 | -8 | -7 | -6 | -5 | -4 | -3 | -2 | -1 |

“text data”

indexing

negative indexing

slicing

substring

upper index excluded

no errors on index out of bounds

immutable

escape sequences:

\n newline

\t tab

\r return

\f form feed

\v vertical tab

string functions:

## Data Structures:

## Lists

ordered

heterogeneous data

allows duplicate data

indexing

negative indexing

slicing

upper index excluded

no errors on index out of bound

**mutable**

nesting

functions:

sort, reverse, append, remove…..

## tuple

ordered

heterogeneous data

allows duplicate data

indexing

negative indexing

slicing

upper index excluded

no errors on index out of bound

tuples by themselves are **immutable**

members can be **mutable**

nesting

functions:

index, count

## sets

unordered

no duplicates

no indexing

no slicing

sets by themselves are **mutable**

members have to be **immutable**

functions:

add, intersection, union

## dictionary

pair of key:value

unordered

indexing is with the key

no slicing

dictionary by themselves are **mutable**

keys:

have to be **immutable**

no duplicates

values:

mutable

duplicates allowed

nesting

functions:

## modules:

1. import mamatha

mamatha.red()

mamatha.white()

mamatha.pink()

1. from mamatha import red, pink

red()

~~white()~~

1. from mamatha import \*

red()

white()

pink()

1. import mamatha as m

m.red()

m.white()

m.pink()

1. from mamatha import red as r

r()

## 

## OOP

Object oriented Programming

C with classes

encapsulation

Polymorphism

Encapsulation