## Python

all codes and exercises at: github.com/shobhit-nigam/snape\_mar



# basics

## language

python.org

### REPL

Read Eval Print Loop

### MISC

data is always initialized

data will always belong to some class

## Data Types

data types are not given

get assigned based on the value

### numbers

int integers

float floating point

complex complex numbers

7 + 8i

7 + 8j

## bool

True

False

### False:

0

0.0

[ ] { } ( )

" "

None

False (bool)

## strings

text data (alphanumeric)

indexed

negative indexed

slicing [x:y]

upper range not included (retrieve till y-1)

no index error

steps

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| t | o | s | h | i | b | a |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 |
| -7 | -6 | -5 | -4 | -3 | -2 | -1 |

immutable

functions:

upper, lower

capitalize, title

### escape sequences

\n new line

\t tab

\b backspace

\r return carriage

\\ \

## comments

# single line

""" '''

used as multiline comments

## naming

just like C/C++/java/C# etc..

\_ numbers alphabets

## operators

### arithmetic

+ add

- sub

\* mul

/ divide

% modulo

// floor division

\*\* power of

### assignment

=

multiple assignment

### compound assignment

+=

-=

….

….

### logical

and

or

not

### relational

<

<=

>

>=

==

!=

### bitwise

&

|

^

~

## keywords

None

True False

if elif else

while

for

## functions

### generic:

print()

type()

len()

id()

input()

### conversion:

str()

int()

float()

complex()

set()

list()

tuple()

sorted()

## Data Structures

## list

any data types

duplicate allowed

sequence

index

negative index

slicing

mutable

functions:

append, insert

remove, pop, clear

sort, reverse

index, count

extend, copy

references

## tuple

any data types

duplicate allowed

sequence

index

negative index

slicing

immutable

functions:

index, count

## set

duplicates not allowed

only unique

sequence

unordered

mutable

data has to be immutable

functions:

add,

## dict

key value pair

sequence

unordered

indexed using keys

keys:

immutable

unique

values:

mutable

dictionary by itself:

mutable

nesting

functions:

## memory

C/C++

|  |  |  |  |
| --- | --- | --- | --- |
| 5680 | varx | ~~40~~ 41 |  |
| 5676 | vary | 60 |  |
| 5672 | varz | 40 |  |
|  |  |  |  |

python

stra = “toshiba”

stra = stra.capitalize()

varz = 40

varz = varz + 1

|  |  |  |  |
| --- | --- | --- | --- |
| 5680 | 40 | varx , ~~varz~~ |  |
| 5676 | 60 | vary |  |
| 5672 | 41 | varz |  |
| 6000 | “toshiba” | ~~stra~~ |  |
| 6010 | “Toshiba” | stra |  |