Basic Data Structures

Python

BASIC DATA STRUCTURES

Name	Type	Description	Example
Integers	int	Whole numbers	1,2,200, 3000
Floating numbers	float	Numbers with decimals	2.3, 4.5, 200.0
Strings	str	Ordered sequence of characters	“Hello”, “3000”, “This is a string”
Lists	list	Ordered sequence of objects	[10, “hello”, 3.44]
Dictionaries	dict	Unordered Key: Value pairs	{“mykey”:”value”, “mykey2”:”value2”}
Tuples	tup	Ordered immutable sequence of objects	(10, “hello”, 3.44)
Sets	set	Unordered collections of unique objects	{“a”, “b”}
Booleans	bool	Logical value	True , False

Numbers

BASIC MATH OPERATIONS

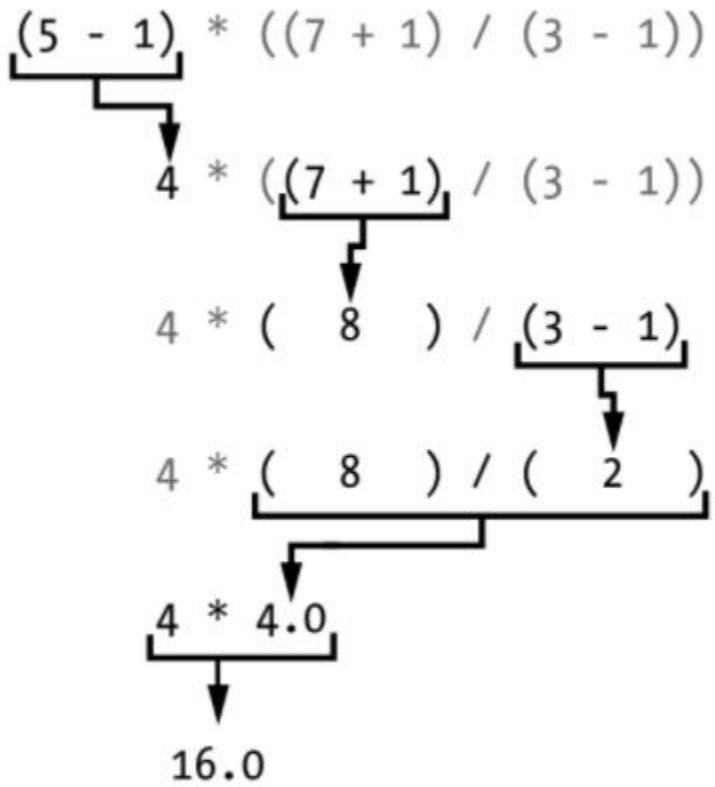
addition	+	subtraction	-
multiplication	*	exponentiation	**
division	/	Floor division (စားလဒ်)	//
Modulus (အကြွင်း)	%		

Operator Precedence

Operators	Meaning
( )	Parentheses
**	Exponent
*, / , // , %	Multiplication, Division, Floor division, Modulus
+, -	Addition, Subtraction
&	Bitwise AND
|	Bitwise OR
==, !=, > , >=, < , <=, is, is not, in , not in	Comparisons, Identity, Membership operators
and	Logical AND
or	Logical OR

- Ordered by operations - ( ) will operate first compared to **

```
>> ( 5 - 1 ) * (( 7 + 1 ) / ( 3 - 1 ))
16.0
```



Rules for variable names

- Names cannot start with a number
- There can be no spaces in the name, use _ instead
- Cannot use any of these symbols

  :'"<>/?|?\()!@#$%^&*~-+
- Avoid using reserved key words in Python

and	else	in	return
as	except	is	True
assert	finally	lambda	try
break	false	nonlocal	with
class	for	None	while
continue	from	not	yield
def	global	or	
del	if	pass	
elif	import	raise	

VARIABLES ASSIGNMENT

- Python uses **“Dynamic Typing”**
- This means you can re-assign variables to different data types
- Different from some other languages that are **“Statically-Typed”**

VARIABLES ASSIGNMENT

```
>> spam = “Hello”
>> spam
‘Hello’

>> spam = “Goodbye”
>> spam
‘Goodbye’
```

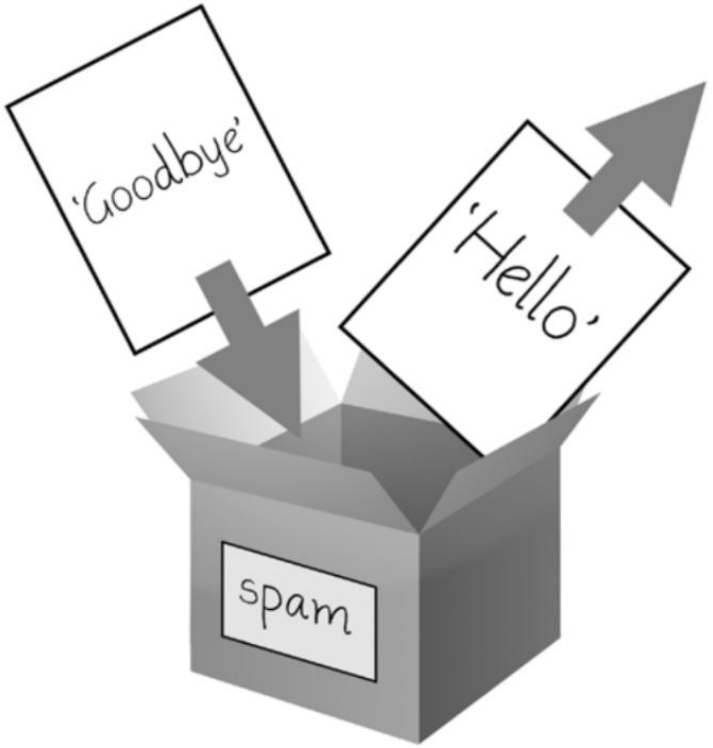


Figure 1-2: When a new value is assigned to a variable, the old one is forgotten.

Strings

Strings

Strings are sequences of characters with the syntax of either single or double quotes:

- ‘Hello world’
- “Hello world”
- “I’m a human”

- Ordered sequences
- Can use indexing and slicing
- Indexing - [int]

Character	**:**	**h**	**e**	**l**	**l**	**o**		**w**	**o**	**r**	**l**	**d**	**!**
Index	**:**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
Reverse Index	**:**	**0**	**-11**	**-10**	**-9**	**-8**	**-7**	**-6**	**-5**	**-4**	**-3**	**-2**	**-1**

- Slicing - [start_int : end_int : step]
- Start_int - numerical index for the slice start
- End_int - numerical index for the slice end (**not include**)
- Step - size of the jump, default is 1

Strings

- Print Formatting with Strings
- Multiple ways to format strings
- Two methods
- **.format()**
- **f-strings**

Checkout the notebook
code_along/01_basic_data_structures/01_basic_and_strings.ipynb

Getting User Inputs

- input(“Text to display”)
- Will get string data type

Lists

Lists

- Ordered sequences
- Can hold a variety of object types

Syntax

- Use **[ ]** and comma **,** to separate objects in list
- Eg - [1 , 2 , 3 , 4 , 5]
  - [‘a’ , ‘b’, ‘c’]
  - [‘this’ , ‘is’, ‘inside’, ‘a’ , ‘list’]
  - [[1 , 2], [1, 2 , 3, [4, 5]]] <- nested list

Go to the notebook
code_along/01_basic_data_structures/02_lists.ipynb

Dictionaries

Dictionaries

- Unordered mapping for storing objects with **key-value pair**
- Can extract without knowing index location

Syntax

- **{‘key1’: ‘value1’,**

  ‘key2’: ‘value2’}

- Value can be any **OBJECTS.**

Dictionaries

- Dictionaries - extract by name (key)
    - Unordered and cannot be sorted
- Lists - extract by index location
    - Ordered sequence and can be sorted and indexed

Go to the notebook
code_along/01_basic_data_structures/03_dictionaries.ipynb

Tuples

Tuples

- Similar to **lists**
- one difference - **immutability**
- It cannot be reassigned.

Syntax

- Parenthesis - ( )
- Eg - (1, 2, 3), (‘a’, ‘b’, ‘c’)

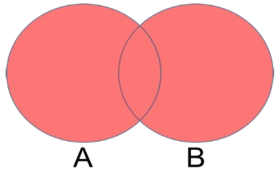
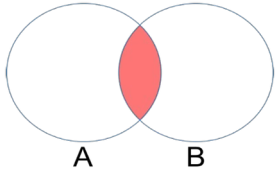
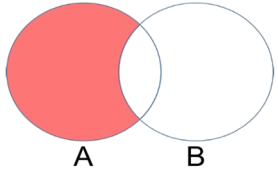
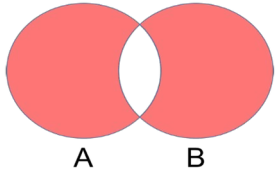
Go to the notebook
code_along/01_basic_data_structures/04_tuple_sets.ipynb

Sets

Sets

- Unordered collections of **unique** elements
- Only on representative of the same object in one set

Sets

Set Operation	Venn Diagram	Interpretation
Union		$A \cup B$, is the set of all values that are a member of A, or B, or both.
Intersection		$A \cap B$, is the set of all values that are members of both A and B.
Difference		$A \setminus B$, is the set of all values of A that are not members of B
Symmetric Difference		$A \triangle B$, is the set of all values which are in one of the sets, but not both.

Go to the notebook
code_along/01_basic_data_structures/04_tuple_sets.ipynb

Booleans

Booleans

- **True or False**
- Very important values with control flow and logic
- Comparison, identity and membership operators will return **boolean values**

== , != , >= , > , <= , < , is , is not , in , not in

Go to the notebook
code_along/01_basic_data_structures/05_booleans.ipynb

Booleans

First Condition	Operator	Second Condition		Result
True	&	True	=>	True
True	&	False	=>	False
False	&	False	=>	False
True	|	True	=>	True
True	|	False	=>	True
False	|	False	=>	False

Go to the notebook
code_along/01_basic_data_structures/05_booleans.ipynb