Data types and programming principles

Thursday, November 12, 2020 12:43 PM

Variables are used store data. We can refer to and manipulate variables in our code.

Data types

Text	str
Numeric	int, float, complex
Sequence	list, tuple
Mapping	dict
Set	set
Boolean	bool
Binary	byte, bytearray, memoryview

Types of variables

Integers	Int, or integer, is a whole number, positive or negative, without decimals, of unlimited length.
Floats	Float, or floating point number is a number, positive or negative, containing one or more decimals.
Complex	Complex numbers are written with a "j" as the imaginary part
Strings	Text defined with quotes
Boolean	true or false
Lists	A collection which is ordered and changeable. Allows duplicate members.
Tuple	A collection which is ordered and unchangeable. Allows duplicate members.
Set	A collection which is unordered and unindexed. No duplicate members
Dictionary	A collection which is unordered, changeable and indexed. No duplicate members.

Naming variables

We have lots of flexibility when naming variables.

We can't start a name with a digit and we have to use alphanumeric characters (A-Z, 0-9) and underscores.

Some names are forbidden. We can't name a variable a keyword

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

You do not need to define the type of variable when you first use the variable as you do in many other languages and you can change the type whenever you like!

Which convention to use?

- **Camel Case**: Second and subsequent words are capitalized, to make word boundaries easier to see. (Presumably, it struck someone at some point that the capital letters strewn throughout the variable name vaguely resemble camel humps.)
  - Example: numberOfCollegeGraduates
- **Cap Case:** Identical to Camel Case, except the first word is also capitalized.
  - Example: NumberOfCollegeGraduates
- **Snake Case:** Words are separated by underscores.
  - Example: number_of_college_graduates

Avoid "l","1","o","O" because these can be easily confused
modules have short all lowercase names. Use underscores if it makes it easier to read
python packages have short all lowercase names but underscores discouraged
variables should be Camel Case and preferring short names
functions should be lowercase with underscores for readability
constants are uppercase with underscores for readability

Strings
We declare these with either single or double quotes.
You can assign a multiline string with three double or single quotes at beginning and end.
Strings are arrays and elements of the array can be accessed with square brackets.

We can slice our string to return a range of characters. Specify the start index and the end index, separated by a colon, to return a part of the string.
Get length of string using len() function
To check if a certain phrase or character is present in a string, we can use the keywords in or not in.
We can convert non-string Python objects to strings using the str function
New line characters "\n" count in the string length
We can access portions of strings via "slicing" [:3]

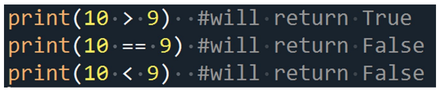
Many useful string methods
- strip() removes any whitespace from the beginning or the end
- lower() returns the string in lower case
- upper() returns the string in upper case
- replace() replaces a string with another string
- split() splits the string into substrings if it finds instances of the separator
- count() returns the number of a given character within a string

To insert characters that are illegal in a string, use an escape character. An escape character is a backslash \ followed by the character you want to insert. An example of an illegal character is a double quote inside a string that is surrounded by double quotes:
Other escape characters used in Python:

Code	**Result**
\'	Single Quote
\\	Backslash
\n	New Line
\r	Carriage Return
\t	Tab
\b	Backspace
\f	Form Feed
\ooo	Octal value
\xhh	Hex value

You can preface a string with r to indicate that characters should be interpreted as is.

Booleans
When you compare two numbers the answer will be given as a boolean



```
print(10 > 9)  #will return True
print(10 == 9) #will return False
print(10 < 9)  #will return False
```

Any value is "true" except 0
Any string is "true" except empty strings
The following will give "False" : (), [], {}, "", the number 0, False, and the value None.
Functions often use booleans to return true or false
Boolean logic involves operators and, or, not

Lists
Lists can contain any and even mixed data types

i	r	o	n	m	a	n
0	1	2	3	4	5	6
-7	-6	-5	-4	-3	-2	-1

i	r	o	n	m	a	n
variable[2:5]						

i	r	o	n	m	a	n
variable[-6:-2]						

first item in a list has index [0], second is [1], third is [2] etc
When slicing the start index is **included**, the stop index is **not included**
lists are ordered meaning entries go in specific order although we can modify these with some list methods.
Adding new items to a list will put it the item at the end.
Lists are changeable (we can add, remove, change etc) after creation
Multiple identical entries are allowed
len() function will return number of items in a list
You can concatenate with "+" sign

Useful methods for lists

- append() will add new list item to end of list
- insert() will insert new item at specified index position

- extend() will append elements from another list (or any other iterable)
- remove() will remove specified item (only first instance)
- pop() removes item at specific index. The del keyword does the same thing.
- clear() empties the list
- sort() sorts the list alphanumerically. To sort descending use keyword argument reverse=true. To sort by custom sort you use key = function
- reverse() will reverse list order
- copy() will create actual new copy of list so you don't modify original. Can also use list() function
- index() will return index of first element of a given value
- count() will return number of instances of a given element

Sets

A collection which is unordered and unindexed. No duplicate members

We write sets with curly brackets {…}. Lists were square brackets

- warning! an empty set cannot be initialized with curly brackets or it will be a dictionary

Sets cannot have multiple entries so they are very helpful for removing duplicates from a list or a tuple.

You can only add something immutable like a string or a tuple. If you try to add a list, you get an error

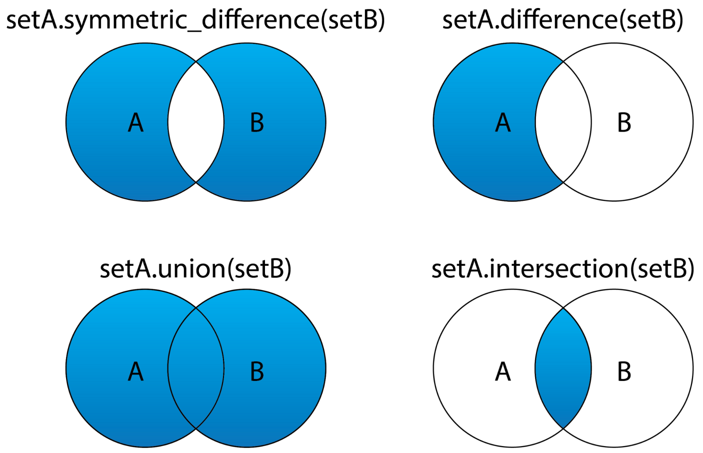
Remove duplicates from a list by making a list of a set. Timing this we see it's more efficient to remove duplicates using a set than via list comprehension (we'll cover this soon)

frozenset() is an immutable set

Useful methods for sets

- add() will add item to set
- remove() will remove item from set *Note: if value is not in set then you get an error, so use discard() and it does the same thing but with no error
- update() will add multiple items to set at a time
- union() will create new set with **all** items in two other sets
- intersection() will create new set with all values in **both** of two other sets
- difference() will create a new set with all values present in one, but not the other.
  - x.difference(y) gives all values of x not present in y
  - also called "relative complement"
- symmetric_difference() gives all values in A, B but **not both** sets.

- issubset() tells you whether all members of B are already members of A making B a subset of A
- isdisjoint() tells you if two sets have no common elements



Tuples

A collection which is ordered and unchangeable. Allows duplicate members.
We write tuples with round brackets (lists were square brackets)
You can also just list comma separated values
We can create nested tuples (a tuple of tuples)
Tuples cannot change their values (immutable) so if you really need to why are you using a tuple? But you can always convert to list, change, then convert list to tuple.
Tuples are used to group together related data, such as a person's name, their age, and their gender.
Tuple assignment is simultaneous rather than sequential making it useful for swapping values
We use tuples when returning multiple values from a function

Dictionaries

In Python dictionaries are written with curly brackets, and they have keys and values
Dictionaries are disordered
Key must be immutable. Scalars like int, float, string are ok. You could use a tuple, but not a list
You can access the items of a dictionary by referring to its key name, inside square brackets:
A dictionary can also contain many dictionaries, this is called nested dictionaries.

Useful methods for dicts

- get() will access items with given key name. You can also include a default value if the key is not there.