1/12/22 Notes

Thursday, January 20, 2022 1:56 PM

1/12/22 Notes

Script

- file -> open new file (in idle python)
- run -> run module
- save in code folder
- F5- interpret & print results

Python Primer 1: Types

Python is an interpreted language

- commands are executed through the Python interpreter
  - interpreter receives a command, evaluates it, and reports the results of the command
- a programmer defines a series of commands in advance and saves those commands in a text file (source code / script)
  - file name ending in **.py**

Example program highlights

- comments: #comments
- white space is important for

Objects in Python

- Python- object oriented language
  - object: instance of a class
- classes form the basis for all data types
- built in classes:
  - **int:** integers
    - 9 = 9 object of class int
  - **float:** floating-point values
    - 9.5 = object of class float
  - **str:** character strings
    - 'hi' = object of class str

Identifiers, Objects, and the Assignment Statement

- Assignment statement
  - most important of all Python commands

    temperature = 98.6
    - establishes temperature as an identifier/name
    - then associates it with the object expressed on the right side of equal sign
      - 98.6: floating-point object, instance of float class
    - ↓temperature:

float 98.6	- identifier pointing to the instance/object of the float class - holds the address of the instance - can be changed to point to another address later

- Alias
  - can establish an **alias** by assigning a second identifier to an existing object

    winter = temperature

↓temperature float 98.6 ↑winter	- both identifiers point to the same address - both are aliases for the same object

- Identifiers
  - case-sensitive
    - temperature is not the same as Temperature
  - can be composed of almost any combination of letters, numerals & underscores
  - CANNOT begin with a numeral
  - CANNOT be a reserved word (33):
    -

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

- Types
  - Python is dynamically typed
    - no advance declaration associating an identifier with a particular data type
      - do not need to declare temperature as a float

```
temperature = 98 #will automatically be type floating-point liter
```

    - identifier has no declared type
      - the object it points to has a definite type
      - temperature is associated with an instance of the float class by having a float value

- Identifier
  - an identifier can be associated with **any type** of object
  - can be later **reassigned** to another object of **any** (the same or different) type

```
temperature = 98.6
# reassign temperature to a new object of a different type
temperature = 99
#OR
temperature = temperature + 5.0
```

- Objects
  - **instantiation**
    - process of creating a new instance/object of a class
    - done by invoking a constructor of a class
      - same as class name

```
#() empty because constructor does not require parameters
w = Widget() #w = object, class name = Widget
```

      - if the constructor did require parameters:

```
w = Widget(a, b, c)
```

    - many of Python's built-in classes are a **literal form** for designing new instances

```
temperature = 98.6 #results in creation of new instance of the float class
```

Classes

- Built-in classes

Class	bool	int	float	list	tuple	str	set	frozenset	dict
Description	boolean value	integer value	floating-point number (decimal)	sequence of objects	sequence of objects	character string	unordered set of distinct objects	unordered set of distinct objects	associative mapping (dictionary)
Immutable?	Yes	Yes	Yes	No	Yes	Yes	No	Yes	No
Use	logical values		integral number, scientific notation						
Instances	True False	whole numbers	decimals 2.0 6.022e23			'' "" """multiple lines"""			
Default constructor	bool() returns false	int() returns 0	float() returns 0.0						

- **immutable class**
  - each object of that class has a **fixed value** upon instantiation that cannot subsequently be changed
  - objects of the float class are immutable
    - cannot be changed to into an int value

- Changing types
  - creation of boolean value from **nonboolean** type using
    bool(foo) **#for value foo/identifier**
    - the interpretation depends on the type of parameter
      - numbers
        - false = 0
        - true = nonzero
      - sequences & other container types (strings, lists)
        - false = empty
        - true = nonempty
  - int() constructor can also construct an integer value based upon an existing value of another type
    - change a value of another type into an int
    - for example
      - if **f** represents a floating-point value
        int(f) **#produces the truncated value of f**
        int(3.14) **# will produce the value 3**
        int(-3.9) **# will produce the value -3**
    - constructor can also be used to parse a string that represents an integer
      int(137) **# will produce the integer value 137**
  - the float() constructor will return the equivalent floating-point value of a parameter
    float(2)#returns 2.0
    float('3.14')#returns 3.14