|  |  |  |
| --- | --- | --- |
| No. | Data | Type |
| 1 | 123 | <class ‘int’> |
| 2 | “this is a great course” | <class ‘str’> |
| 3 | 24.75 | <class ‘float’> |
| 4 | [“try”,“me”,“out”] | <class ‘list’> |
| 5 | {2, ‘a’, 3, “c”} | <class ‘set’> |

Q2(b)

TheplaypeopleGames

Q3.(a)

[0, 3, 6, 9, 12, 15]

Q3.(b)

immutable types: int, float, long, str, tuple

mutable types: list, dict, set

Q4(a)

Import os.path as a module to use

Q4(b)

os.path works in a funny way. It looks like os should be a package with a submodule path, but in reality os is a normal module that does magic with sys.modules to inject os.path. Here's what happens:

* When Python starts up, it loads a bunch of modules into sys.modules. They aren't bound to any names in your script, but you can access the already-created modules when you import them in some way.
  + sys.modules is a dict in which modules are cached. When you import a module, if it already has been imported somewhere, it gets the instance stored in sys.modules.
* os is among the modules that are loaded when Python starts up. It assigns its path attribute to an os-specific path module.
* It injects sys.modules['os.path'] = path so that you're able to do "import os.path" as though it was a submodule.

I tend to think of os.path as a module I want to use rather than a thing in the*os*module, so even though it's not really a submodule of a package called os, I import it sort of like it is one and **I always do import os.path**. This is consistent with how os.path is documented.

documented.

Q5 (a)

def myPrint( str ):

print (str)

return

(b)

keyword arguments: Call a function with parameters by passing arguments preceded by an identifier (e.g. name=) which should be same as the name of parameter in a function. Or passed as a value in a dictionary preceded by \*\*.

Positional arguments: Call a function with parameters by passing arguments without preceding by an identifier (e.g. name=) in a function, or be passed as elements of an interable preceded by \*.

(c)

def foo(a,b):

print (a+b)

return

Example via keywords arguments,

foo (a = 1, b =2)

foo (\*\*{‘a’:1, ‘b’:2})

Example via positional arguments,

foo (1, 2)

foo (\*(1, 2))

Q6(a).

Q6(b).

def perfect\_interleave(lst1,lst2,lst3):

if len(lst1) != len(lst2) or len(lst1) != len(lst3):

raise Exception('Should input same size list')

elif len(lst1) == 0 or len(lst2) == 0 or len(lst3) == 0 :

return []

else:

return [lst1[0],lst2[0],lst3[0]] + perfect\_interleave(lst1[1:],lst2[1:],lst3[1:])

Q8(a)

1. 1.6666666666666667
2. 23
3. 29
4. False
5. not ((4.5 < 12.9) and (6 \* 2 <= 13))
6. True

(b) A and E