http://www.mypythonquiz.com/question.php?qid=259

Question #1: **what does the following code do?**

```
def a(b, c, d): pass
```

- defines a list and initializes it
- defines a function, which does nothing - **correct**
- defines a function, which passes its parameters through
- defines an empty class

description: The 'def' statement defines a function. The 'pass' statement is a null operation.

Question #2: **what gets printed? Assuming python version 2.x**
54% on 12514 times asked

```
print type(1/2)
```

- <type 'int'> **- correct**
- <type 'number'>
- <type 'float'>
- <type 'double'>
- <type 'tuple'>

description: division of an integer by another integer yelds an integer in version 2.x of python

Question #3: **what is the output of the following code?**
75% on 8631 times asked

```
print type([1,2])
```

- <type 'tuple'>
- <type 'int'>
- <type 'set'>
- <type 'complex'>
- <type 'list'> **- correct**

description: Lists are formed by placing a comma-separated list of expressions in square brackets

Question #4: **what gets printed?**
55% on 10233 times asked

```
def f(): pass
print type(f())
```

- <type 'function'>
- <type 'tuple'>
- <type 'NoneType'> - **correct**
- <type 'str'>
- <type 'type'>

description: The argument to the type() call is a return value of a function call, which returns None

Question #5: **what should the below code print?**
69% on 7845 times asked

```
print type(1J)
```

- <type 'complex'> - **correct**
- <type 'unicode'>
- <type 'int'>
- <type 'float'>
- <type 'dict'>

description: An imaginary literal yields a complex number with a real part of 0.0

Question #6: **what is the output of the following code?**
51% on 9024 times asked

```
print type(lambda:None)
```

- <type 'NoneType'>
- <type 'tuple'>
- <type 'type'>
- <type 'function'> - **correct**
- <type 'bool'>

description: 'lambda arguments: expression' yields a function object

Question #7: **what is the output of the below program?**
58% on 9074 times asked

```
a = [1,2,3,None,(),[],]
print len(a)
```

- syntax error
- 4
- 5
- 6 - **correct**
- 7

description: The trailing comma in the list is ignored, the rest are legitimate values

Question #11: **What gets printed?**
55% on 7488 times asked

```
nums = set([1,1,2,3,3,3,4])
print len(nums)
```

- 1
- 2
- 4 - **correct**
- 5
- 7

description: nums is a set, so only unique values are retained.

Question #12: **What gets printed?**
60% on 6427 times asked

```
x = True
y = False
z = False

if x or y and z:
    print "yes"
else:
    print "no"
```

- yes - **correct**
- no

fails to compile

description: AND is higher precedence than OR in python and is evaluated first

Question #14: **If PYTHONPATH is set in the environment, which directories are searched for modules?**
49% on 5750 times asked

```
A) PYTHONPATH directory

B) current directory

C) home directory

D) installation dependent default path
```

A only

A and D

A, B, and C

A, B, and D - **correct**

A, B, C, and D

description: First is the current directory, then is the PYTHONPATH directory if set, then is the installation dependent default path

Question #15: **In python 2.6 or earlier, the code will print error type 1 if accessSecureSystem raises an exception of either AccessError type or SecurityError type**
49% on 3462 times asked

```
try:
  accessSecureSystem()
except AccessError, SecurityError:
  print "error type 1"

continueWork()
```

true

false - **correct**

description: The except statement will only catch exceptions of type AccessError and name the exception object SecurityError. In order to catch both you can use a tuple like this: except (AccessError, SecurityError). Python has been changed in version 3.0 so that the syntax shown in the question will actually catch both types.

Question #16: **The following code will successfully print the days and then the months**
60% on 3573 times asked

```
daysOfWeek = ['Monday',
              'Tuesday',
              'Wednesday',
              'Thursday',
              'Friday',
              'Saturday',
              'Sunday']

months =             ['Jan', \
                      'Feb', \
                      'Mar', \
                      'Apr', \
                      'May', \
                      'Jun', \
                      'Jul', \
                      'Aug', \
                      'Sep', \
                      'Oct', \
                      'Nov', \
                      'Dec']

print "DAYS: %s, MONTHS %s" %
    (daysOfWeek, months)
```

- true
- false - **correct**

description: daysOfWeek is ok because expressions in parentheses, square brackets or curly braces can be split over more than one physical line without using backslashes. months is ok because backslashes are used to join physical lines, even though they are not required in this case. The print statement will not print the data because 2 logical lines are used without a backslash to join them into a logical line.

Question #17: **Assuming python 2.6 what gets printed?**
51% on 4206 times asked

```
f = None

for i in range (5):
    with open("data.txt", "w") as f:
        if i > 2:
            break

print f.closed
```

- True - **correct**
- False
- None

description: The WITH statement when used with open file guarantees that the file object is closed when the with block exits.

Question #18: **What gets printed?**
55% on 4502 times asked

```
counter = 1

def doLotsOfStuff():

    global counter

    for i in (1, 2, 3):
        counter += 1

doLotsOfStuff()

print counter
```

- 1
- 3
- 4 - **correct**
- 7
- none of the above

description: the counter variable being referenced in the function is the global variable defined outside of the function. Changes to the variable in the function affect the original variable.

Question #19: **What gets printed?**
55% on 4401 times asked

```
print r"\nwoow"
```

- new line then the string: woow
- the text exactly like this: r"\nwoow"
- the text like exactly like this: \nwoow - **correct**
- the letter r and then newline then the text: woow
- the letter r then the text like this: nwoow

description: When prefixed with the letter 'r' or 'R' a string literal becomes a raw string and the escape sequences such as \n are not converted.

Question #20: **What gets printed?**

58% on 4279 times asked

```
print "hello" 'world'
```

- on one line the text: hello world
- on one line the text: helloworld - **correct**
- hello on one line and world on the next line
- syntax error, this python program will not run

description: String literals seperated by white space are allowed. They are concatenated.

Question #21: **What gets printed?**
53% on 4458 times asked

```
print "\x48\x49!"
```

- \x48\x49!
- 4849
- 4849!
- 48 49!
- HI! - **correct**

description: \x is an escape sequence that means the following 2 digits are a hexadicmal number encoding a character.

Question #22: **What gets printed?**
63% on 3171 times asked

```
print 0xA + 0xa
```

- 0xA + 0xa
- 0xA 0xa
- 14
- 20 - **correct**
- 0x20

description: 0xA and 0xa are both hexadecimal integer literals representing the decimal value 10. There sum is 20.

Question #24: **What gets printed?**

69% on 3313 times asked

```
kvps  = {"user","bill", "password","hillary"}

print kvps['password']
```

- user
- bill
- password
- hillary
- Nothing. Python syntax error - **correct**

description: When initializing a dictionary, key and values are seperated by colon and key-value pairs are separated by commas.
kvps = {"user":"bill", "password":"hillary"}

Question #25: **What gets printed?**
64% on 3259 times asked

```
class Account:
    def __init__(self, id):
        self.id = id
        id = 666

acc = Account(123)
print acc.id
```

- None
- 123 - **correct**
- 666
- SyntaxError, this program will not run

description: class instantiation automatically calls the __init__ method and passes the object as the self parameter. 123 is assigned to data attribute of the object called id. The 666 value is not retained in the object as it is not assigned to a data attribute of the class/object.

Question #26: **What gets printed?**
56% on 3709 times asked

```
name = "snow storm"

print "%s" % name[6:8]
```

- st
- sto
- to - **correct**
- tor
- Syntax Error

description: This is a slice of a string from index 6 to index 8 not including index 8. The first character in the string is position 0.

Question #27: **What gets printed?**
54% on 3331 times asked

```
name = "snow storm"

name[5] = 'X'

print name
```

- snow storm
- snowXstorm
- snow Xtorm
- ERROR, this code will not run - **correct**

description: TypeError. You can not modify the contents of a string

Question #28: **Which numbers are printed?**
62% on 3470 times asked

```
for i in  range(2):
    print i

for i in range(4,6):
    print i
```

- 2, 4, 6
- 0, 1, 2, 4, 5, 6
- 0, 1, 4, 5 - **correct**
- 0, 1, 4, 5, 6, 7, 8, 9
- 1, 2, 4, 5, 6

description: If only 1 number is supplied to range it is the end of the range. The default beginning of a range is 0. The range will include the beginning of the range and all numbers up to but not including the end of the range.

Question #30: **What sequence of numbers is printed?**
70% on 2504 times asked

```
values = [2, 3, 2, 4]

def my_transformation(num):
    return num ** 2

for i in  map(my_transformation, values):
    print i
```

- 2 3 2 4
- 4 6 4 8
- 1 1.5 1 2
- 1 1 1 2
- 4 9 4 16 - **correct**

description: map will call the function for each value in the list. The ** operator in the function raises the parameter to the power of 2.

Question #31: **What numbers get printed**
67% on 1990 times asked

```
import pickle

class account:
        def __init__(self, id, balance):
                self.id = id
                self.balance = balance
        def deposit(self, amount):
                self.balance += amount
        def withdraw(self, amount):
                self.balance -= amount

myac = account('123', 100)
myac.deposit(800)
myac.withdraw(500)

fd = open( "archive", "w" )
pickle.dump( myac, fd)
fd.close()

myac.deposit(200)
print myac.balance

fd = open( "archive", "r" )
myac = pickle.load( fd )
fd.close()
```

```
print myac.balance
```

- 500 300
- 500 500
- 600 400 - **correct**
- 600 600
- 300 500

description: pickle will store the state of the account object to file when its value is 400. After storing the value to file 200 is added and 600 is printed. After printing 600 the object form file is reloaded from file and printed with a value of 400.

Question #32: **What gets printed by the code snippet below?**
53% on 2621 times asked

```
import math

print math.floor(5.5)
```

- 5
- 5.0 - **correct**
- 5.5
- 6
- 6.0

description: the floor method will return the largest integer value less than or equal to the parameter as a float type.

Question #33: **What gets printed by the code below?**
54% on 1987 times asked

```
class Person:
    def __init__(self, id):
        self.id = id

obama = Person(100)

obama.__dict__['age'] = 49

print obama.age + len(obama.__dict__)
```

- 1

- 2
- 49
- 50
- 51 - **correct**

description: We have created a member variable named 'age' by adding it directly the objects dictionary. The value of 'age' is initialized to 49. There are 2 items in the dictionary, 'age' and 'id', therefore the sum of the 'age' value 49 and then size of the dictionary, 2 items, is 51.

Question #34: **What gets printed?**
66% on 2077 times asked

```
x = "foo "
y = 2
print x + y
```

- foo
- foo foo
- foo 2
- 2
- An exception is thrown - **correct**

description: Python is a strongly typed language. Once a variable has a type, it must be casted to change the type. x is a string and y is an integer. Trying to concatenate them will cause an exception of type TypeError

Question #36: **What does the code below do?**
62% on 1874 times asked

```
sys.path.append('/root/mods')
```

- Changes the location that the python executable is run from
- Changes the current working directory
- Adds a new directory to seach for python modules that are imported - **correct**
- Removes all directories for mods
- Changes the location where sub-processes are searched for after they are launched

description: The list sys.path contains, in order, all the directories to be searched when trying to load a module

Question #37: **What gets printed?**

45% on 2385 times asked

```
import re
sum = 0

pattern = 'back'
if re.match(pattern, 'backup.txt'):
    sum += 1
if re.match(pattern, 'text.back'):
    sum += 2
if re.search(pattern, 'backup.txt'):
    sum += 4
if re.search(pattern, 'text.back'):
    sum += 8

print sum
```

- 3
- 7
- 13 - **correct**
- 14
- 15

description: search will see if the pattern exists anywhere in the string, while match will only check if the pattern exists in the beginning of the string.

Question #38: **Which of the following print statements will print all the names in the list on a seperate line**
56% on 2178 times asked

```
names = ['Ramesh', 'Rajesh', 'Roger', 'Ivan', 'Nico']
```

- print "\n".join(names) - **correct**
- print names.join("\n")
- print names.concatenate("\n")
- print names.append("\n")
- print names.join("%s\n", names)

description: Only A is valid syntax. There is a join method to string objects which takes an iterable object as parameter and combines the string calling the method in between each item to produce a resulting string.

Question #41: **What gets printed**
74% on 1604 times asked

```
foo = {}
print type(foo)
```

- set
- dict - **correct**
- list
- tuple
- object

description: Curly braces are the syntax for a dictionary declaration

Question #42: **What gets printed?**
71% on 1603 times asked

```
foo = (3, 4, 5)
print type(foo)
```

- int
- list
- tuple - **correct**
- dict
- set

description: Parentheses are used to initialize a tuple.

Question #43: **What gets printed?**
65% on 1713 times asked

```
country_counter = {}

def addone(country):
    if country in country_counter:
        country_counter[country] += 1
    else:
        country_counter[country] = 1

addone('China')
addone('Japan')
addone('china')

print len(country_counter)
```

- 0
- 1
- 2
- 3 - **correct**
- 4

description: The len function will return the number of keys in a dictionary. In this case 3 items have been added to the dictionary. Note that the key's to a dictionary are case sensitive.

Question #44: **What gets printed?**
62% on 1756 times asked

```
confusion = {}
confusion[1] = 1
confusion['1'] = 2
confusion[1] += 1

sum = 0
for k in confusion:
    sum += confusion[k]

print sum
```

- 1
- 2
- 3
- 4 - **correct**
- 5

description: Note that keys to a dictionary can be mixed between strings and integers and they represent different keys.

Question #48: **What gets printed?**
67% on 1523 times asked

```
foo = {1:'1', 2:'2', 3:'3'}
foo = {}
print len(foo)
```

- 0 - **correct**
- 1
- 2
- 3
- An exception is thrown

description: after the second line of code, foo is an empty dictionary. The proper way to

actually remove all items from a dictionary is to call the 'clear' method of the dictionary object

Question #43: **What gets printed?**
65% on 1712 times asked

```
country_counter = {}

def addone(country):
    if country in country_counter:
        country_counter[country] += 1
    else:
        country_counter[country] = 1

addone('China')
addone('Japan')
addone('china')

print len(country_counter)
```

- 0
- 1
- 2
- 3 **- correct**
- 4

description: The len function will return the number of keys in a dictionary. In this case 3 items have been added to the dictionary. Note that the key's to a dictionary are case sensitive.