Question #1: **what does the following code do?**
74% on 9963 times asked

```
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**
55% on 6422 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?**
77% on 4412 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 5293 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?**
70% on 4017 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?**
52% on 4528 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?**
59% on 4651 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 #8: **what gets printed? Assuming python version 3.x**
48% on 4114 times asked

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

○ <type 'int'>

○ <type 'number'>

○ <type 'float'> - **correct**

○ <type 'double'>

○ <type 'tuple'>

description: division of an integer by another integer yelds a float in version 3.x of python. Also note there is a changed print syntax in python 3.

Question #9: **What gets printed?**
81% on 2627 times asked

```
d = lambda p: p * 2
t = lambda p: p * 3
x = 2
x = d(x)
x = t(x)
x = d(x)
print x
```

○ 7

○ 12

○ 24 – **correct**

○ 36

○ 48

Description: start with 2, multiply by 2, multiply by 3, multipy by 2.

Question #10: **What gets printed?**
64% on 3409 times asked

```
x = 4.5
y = 2
print x//y
```

- ○ 2.0 – **correct**
- ○ 2.25
- ○ 9.0
- ○ 20.25
- ○ 21

description: this is truncating division. The remainder is dropped.

Question #11: **What gets printed?**
55% on 3792 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?**
59% on 3256 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 #13: **What gets printed?**
54% on 3068 times asked

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

if not x or y:
    print 1
elif not x or not y and z:
    print 2
elif not x or y or not y and x:
    print 3
else:
    print 4
```

○ 1

○ 2

○ 3 – **correct**

○ 4

description: NOT has first precedence, then AND, then OR

Question #14: **If PYTHONPATH is set in the environment, which directories are searched for modules?**
49% on 2680 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**
48% on 1641 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**
61% on 1709 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?**
50% on 2007 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 2122 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?**
57% on 2019 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 2021 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?**
56% on 1997 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?**
65% on 1454 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?**
70% on 1543 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?**
66% on 1507 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?**
59% on 1710 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?**
55% on 1576 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 1669 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 #29: **What sequence of numbers is printed?**
52% on 1516 times asked

```
values = [1, 2, 1, 3]
nums = set(values)

def checkit(num):
    if num in nums:
        return True
    else:
        return False

for i in  filter(checkit, values):
    print i
```

- ○ 1 2 3
- ○ 1 2 1 3 – **correct**
- ○ 1 2 1 3 1 2 1 3
- ○ 1 1 1 1 2 2 3 3
- ○ Syntax Error

description: The filter will return all items from the list values which return True when passed to the function checkit. checkit will check if the value is in the set. Since all the numbers in the set come from the values list, all of the orignal values in the list will return True.

Question #30: **What sequence of numbers is printed?**
73% on 1111 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**
68% on 925 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?**
54% on 973 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?**
52% on 778 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?**
68% on 779 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 #35: **What gets printed?**
76% on 613 times asked

```
def simpleFunction():
    "This is a cool simple function that returns 1"
    return 1

print simpleFunction.__doc__[10:14]
```

○ simpleFunction

○ simple

○ func

○ funtion

○ cool - **correct**

description: There is a docstring defined for this method, by putting a string on the first line after the start of the function definition. The docstring can be referenced using the __doc__ attribute of the function.

Question #36: **What does the code below do?**
63% on 680 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 859 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 807 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 #39: **True or false? Code indentation must be 4 spaces when creating a code block?**
66% on 547 times asked

```
if error:
    # four spaces of indent are used to create the block
    print "%s" % msg
```

- ○ True
- ○ False - **correct**

description: This is false. Indentation needs to be consistent. A specific number of spaces used for indentation is not prescribed by the language.

Question #40: **Assuming the filename for the code below is /usr/lib/python/person.py and the program is run as:**
`python /usr/lib/python/person.py`

What gets printed?
47% on 790 times asked

```
class Person:
    def __init__(self):
        pass

    def getAge(self):
        print __name__

p = Person()
p.getAge()
```

○ Person

○ getAge

○ usr.lib.python.person

○ __main__ - **correct**

○ An exception is thrown

description: If the module where the reference to __name__ is made has been imported from another file, then the module name will be in the variable in the form of the filename without the path or file extension. If the code is being run NOT as the result of an import, the variable will have the special value "__main__".

Question #41: **What gets printed**
77% on 592 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?**
76% on 567 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?**
67% on 625 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?**
64% on 617 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 #45: **What gets printed?**
42% on 800 times asked

```
confusion = {}
confusion[1] = 1
confusion['1'] = 2
confusion[1.0] = 4

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

print sum
```

- ○ 2
- ○ 4
- ○ 6 – **correct**
- ○ 7
- ○ An exception is thrown

description: Note from python docs: "if two numbers compare equal (such as 1 and 1.0) then they can be used interchangeably to index the same dictionary entry. (Note however, that since computers store floating-point numbers as approximations it is usually unwise to use them as dictionary keys.)"

Question #46: **What gets printed?**
45% on 805 times asked

```
boxes = {}
jars = {}
crates = {}

boxes['cereal'] = 1
boxes['candy'] = 2
jars['honey'] = 4
crates['boxes'] = boxes
crates['jars'] = jars

print len(crates[boxes])
```

- 1
- 2
- 4
- 7
- An exception is thrown - **correct**

description: Keys can only be immutable types, so a dictionary can not be used as a key. In the print statement the dictionary is used as the key instead of the string 'boxes'. Had the string been used it would have printed the length of the boxes dictionary which is 2.

Question #47: **What gets printed?**
61% on 556 times asked

```
numberGames = {}
numberGames[(1,2,4)] = 8
numberGames[(4,2,1)] = 10
numberGames[(1,2)] = 12

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

print len(numberGames) + sum
```

- 8
- 12
- 24
- 30
- 33 - **correct**

description: Tuples can be used for keys into dictionary. The tuples can have mixed length and the order of the items in the tuple is considered when comparing the equality of the keys.

Question #48: **What gets printed?**
69% on 564 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 #49: **What gets printed?**
70% on 513 times asked

```
foo = {1:'1', 2:'2', 3:'3'}
del foo[1]
foo[1] = '10'
del foo[2]
print len(foo)
```

- 1
- 2 – **correct**
- 3
- 4
- An exception is thrown

description: The del function is used to remove key value pairs from a dictionary.

Question #50: **What gets printed?**
78% on 552 times asked

```
names = ['Amir', 'Barry', 'Chales', 'Dao']
print names[-1][-1]
```

- A
- r
- Amir
- Dao
- o - **correct**

description: -1 refers to the last position in a list or the last character in a string. In this case, we are referencing the last character in the last string in the list.

Question #51: **What gets printed?**
51% on 598 times asked

```
names1 = ['Amir', 'Barry', 'Chales', 'Dao']
names2 = names1
names3 = names1[:]

names2[0] = 'Alice'
names3[1] = 'Bob'

sum = 0
for ls in (names1, names2, names3):
    if ls[0] == 'Alice':
        sum += 1
    if ls[1] == 'Bob':
        sum += 10

print sum
```

- 11
- 12 – **correct**
- 21
- 22
- 33

description: When assigning names1 to names2, we create a second reference to the same list. Changes to names2 affect names1. When assigning the slice of all elements in names1 to names3, we are creating a full copy of names1 which can be modified independently.

Question #52: **What gets printed?**
68% on 514 times asked

```
names1 = ['Amir', 'Barry', 'Chales', 'Dao']

loc = names1.index("Edward")

print loc
```

- -1
- 0
- 4
- Edward
- An exception is thrown - **correct**

description: If index can not find the specified value in the list an exception is thrown.

Question #53: **What gets printed?**
82% on 489 times asked

```
names1 = ['Amir', 'Barry', 'Chales', 'Dao']

if 'amir' in names1:
    print 1
else:
    print 2
```

- ○ 1
- ○ 2 – **correct**
- ○ An exception is thrown

description: the in keyword can be used to search for a value in a list, set, or dict. In this case the search fails, because the string value is case sensitive.

Question #54: **What gets printed?**
76% on 491 times asked

```
names1 = ['Amir', 'Barry', 'Chales', 'Dao']
names2 = [name.lower() for name in names1]

print names2[2][0]
```

- ○ i
- ○ a
- ○ c – **correct**
- ○ C
- ○ An exception is thrown

description: List Comprehensions are a shorthand to creating a new list with the all the values in a original list modified by some python expression.

Question #55: **What gets printed?**
61% on 624 times asked

```
numbers = [1, 2, 3, 4]
numbers.append([5,6,7,8])
print len(numbers)
```

- ○ 4
- ○ 5 – **correct**
- ○ 8
- ○ 12
- ○ An exception is thrown

description: When a list is passed to the append method of list, the entire list is added as an element of the list. The lists are not merged.

Question #56: **Which of the following data structures can be used with the "in" operator to check if an item is in the data structure?**
69% on 516 times asked

- ○ list
- ○ set
- ○ dictionary
- ○ None of the above
- ○ All of the above - **correct**

description: The "in" operator can be used with all 3 of these data structures.

Question #57: **Wat gets printed?**
78% on 476 times asked

```
list1 = [1, 2, 3, 4]
list2 = [5, 6, 7, 8]

print len(list1 + list2)
```

- ○ 2
- ○ 4
- ○ 5
- ○ 8 – **correct**
- ○ An exception is thrown

description: The + operator appends the elements in each list into a new list

Question #58: **What gets printed?**
70% on 509 times asked

```
def addItem(listParam):
    listParam += [1]

mylist = [1, 2, 3, 4]
addItem(mylist)
print len(mylist)
```

- ○ 1
- ○ 4
- ○ 5 – **correct**
- ○ 8
- ○ An exception is thrown

description: The list is passed by reference to the function and modifications to the function parameter also effect the original list.

Question #59: **What gets printed?**
53% on 573 times asked

```
my_tuple = (1, 2, 3, 4)
my_tuple.append( (5, 6, 7) )
print len(my_tuple)
```

- ○ 1
- ○ 2
- ○ 5
- ○ 7
- ○ An exception is thrown - **correct**

description: Tuples are immutable and don't have an append method. An exception is thrown in this case.

Question #60: **What gets printed?**
78% on 477 times asked

```
a = 1
b = 2
a,b = b,a

print "%d %d" % (a,b)
```

- ○ 1 2
- ○ 2 1 – **correct**
- ○ An exception is thrown
- ○ This program has undefined behavior

description: This is valid python code. This is assignment multiple variables at once. The values in b and a are being assigned to a and be respectively.

Question #61: **What gets printed?**
73% on 385 times asked

```
def print_header(str):
    print "+++%s+++" % str

print_header.category = 1
print_header.text = "some info"
print_header("%d %s" %  \
(print_header.category, print_header.text))
```

○ +++1 some info+++ - **correct**

○ +++%s+++

○ 1

○ 1

○ some info

description: As of python 2.1 you could assign arbitrary typed information to functions.

Question #62: **What gets printed?**
58% on 507 times asked

```
def dostuff(param1, *param2):
   print type(param2)

dostuff('apples', 'bananas', 'cherry', 'dates')
```

○ str

○ int

○ tuple – **correct**

○ list

○ dict

description: param2 aggregates remaining parameters into a tuple.

Question #63: **What gets printed?**
69% on 415 times asked

```
def dostuff(param1, **param2):
   print type(param2)


dostuff('capitals', Arizona='Phoenix',
California='Sacramento', Texas='Austin')
```

- in
- str
- tuple
- list
- dict - **correct**

description: param2 aggregates the remaining parameters into a dictionary.

Question #64: **What gets printed?**
65% on 479 times asked

```
def myfunc(x, y, z, a):
    print x + y
nums = [1, 2, 3, 4]

myfunc(*nums)
```

- 1
- 3 – **correct**
- 6
- 10
- An exception is thrown

description: *nums will unpack the list into individual elements to be passed to the function.

Question #65: **How do you create a package so that the following reference will work?**
45% on 552 times asked

```
p = mytools.myparser.MyParser()
```

- Declare the myparser package in mytools.py
- Create an __init__.py in the home dir
- Inside the mytools dir create a __init__.py – **correct**
- Create a myparser.py directory inside the mytools directory
- This can not be done

description: In order to create a package create a directory for the package name and then put an __init__.py file in te directory.

Question #66: **What gets printed?**
58% on 440 times asked

```
class A:
    def __init__(self, a, b, c):
        self.x = a + b + c

a = A(1,2,3)
b = getattr(a, 'x')
setattr(a, 'x', b+1)
print a.x
```

- ○ 1
- ○ 2
- ○ 3
- ○ 6
- ○ 7 - **correct**

description: getattr can be used to get the value of a member variable of an object. setattr can be used to set it.

Question #67: **What gets printed?**
41% on 393 times asked

```
class NumFactory:
    def __init__(self, n):
        self.val = n
    def timesTwo(self):
        self.val *= 2
    def plusTwo(self):
        self.val += 2

f = NumFactory(2)
for m in dir(f):
    mthd = getattr(f,m)
    if callable(mthd):
        mthd()

print f.val
```

- ○ 2
- ○ 4
- ○ 6
- ○ 8
- ○ An exception is thrown - **correct**

description: An exception will be thrown when trying to call the __init__ method of the object without any parameters: TypeError: __init__() takes exactly 2 arguments (1 given)

Question #68: **What gets printed?**
75% on 374 times asked

```
one = chr(104)
two = chr(105)
print "%s%s" % (one, two)
```

- ○ hi – **correct**
- ○ h
- ○ Inside the mytools dir create a __init__.py and myparser.py
- ○ 104105
- ○ 104

description: char is a built in function that converts an ascii code to a 1 letter string.

Question #69: **What gets printed?**
63% on 394 times asked

```
x = 0
y = 1

a = cmp(x,y)
if a < x:
    print "a"
elif a == x:
    print "b"
else:
    print "c"
```

- ○ a – **correct**
- ○ b
- ○ c

description: cmp returns a value less than 0 if x is less than y.
cmp returns 0 if x equals y.
cmp returns a value greater than 0 if x is greater than y.

Question #70: **What gets printed?**
60% on 413 times asked

```
x = 1
y = "2"
z = 3

sum = 0
for i in (x,y,z):
    if isinstance(i, int):
        sum += i
print sum
```

○ 2

○ 3

○ 4 – **correct**

○ 6

○ An exception is thrown

description: isinstance will return true if the first parameter is an instance of the class type of the second parameter.

Question #71: **What gets printed (with python version 2.X) assuming the user enters the following at the prompt?**
#: foo
45% on 459 times asked

```
a = input("#: ")

print a
```

○ f

○ foo

○ #: foo

○ An exception is thrown - **correct**

description: The input function is equivalent to:
eval(raw_input(prompt)) This function will attempt to execute the text tempted at the prompt as python code. In the case of this input, invalid python code was entered an exception is thrown

Question #72: **What gets printed?**
69% on 467 times asked

```
x = sum(range(5))
print x
```

- ○ 4
- ○ 5
- ○ 10 – **correct**
- ○ 15
- ○ An exception is thrown

description: range(5) produces a list of the numbers 0, 1, 2, 3, 4.
sum will add all the numbers in the list.

Question #73: **If the user types '0' at the prompt what gets printed?**
58% on 370 times asked

```
def getinput():
    print "0: start"
    print "1: stop"
    print "2: reset"
    x = raw_input("selection: ")
    try:
        num = int(x)
        if num > 2 or num < 0:
            return None
        return num
    except:
        return None

num = getinput()
if not num:
    print "invalid"
else:
    print "valid"
```

- ○ valid
- ○ invalid – **correct**
- ○ An exception is thrown

description: 0 is returned from getinput. Remember that both 0, None, empty sequences and some other forms all evaluate to False in truth testing.

Question #74: **What gets printed?**
72% on 398 times asked

```
kvps = { '1' : 1, '2' : 2 }
theCopy = kvps

kvps['1'] = 5

sum = kvps['1'] + theCopy['1']
print sum
```

- ○ 1
- ○ 2
- ○ 7
- ○ 10 – **correct**
- ○ An exception is thrown

description: Assignment will provide another reference to the same dictionary. Therefore the change to the original dictionary also is changing the copy.

Question #75: **What gets printed?**
77% on 359 times asked

```
kvps = { '1' : 1, '2' : 2 }
theCopy = kvps.copy()

kvps['1'] = 5

sum = kvps['1'] + theCopy['1']
print sum
```

- ○ 1
- ○ 2
- ○ 6 – **correct**
- ○ 10
- ○ An exception is thrown

description: The copy method of the dictionary will make a new (shallow) copy of the dictionary so a change to the original in this case does not change the copy.

Question #76: **What gets printed**
49% on 533 times asked

```
aList = [1,2]
bList = [3,4]

kvps = { '1' : aList, '2' : bList }
theCopy = kvps.copy()

kvps['1'][0] = 5

sum = kvps['1'][0] + theCopy['1'][0]
print sum
```

- ○ 1
- ○ 2
- ○ 6
- ○ 10 – **correct**
- ○ An exception is thrown

description: The copy method provides a shallow copy therefore the list being held as the value inside the dictionary is the same list in the copy as the original.

Question #77: **What gets printed?**
83% on 389 times asked

```
import copy

aList = [1,2]
bList = [3,4]

kvps = { '1' : aList, '2' : bList }
theCopy = copy.deepcopy(kvps)

kvps['1'][0] = 5

sum = kvps['1'][0] + theCopy['1'][0]
print sum
```

- ○ 1
- ○ 2
- ○ 6 – **correct**
- ○ 10
- ○ An exception is thrown

description: A deep copy will copy all the keys and values inside a dictionary. Therefore the list inside the dictionary are different in the first and second dictionaries of this example.

Question #78: **What gets printed?**
61% on 395 times asked

```
kvps = { '1' : 1, '2' : 2 }
theCopy = dict(kvps)

kvps['1'] = 5

sum = kvps['1'] + theCopy['1']
print sum
```

- ○ 1
- ○ 2
- ○ 6 – **correct**
- ○ 10
- ○ An exception is thrown

description: Creating a new dictionary object initialized from the first does a 'shallow copy'

Question #79: **What gets printed?**
86% on 371 times asked

```
kvps = { '1' : 1, '2' : 2 , '3' : 3, '4' : 4, '5' : 5}
newData = { '1' : 10, '3' : 30 }

kvps.update(newData)

x = sum(kvps.values())

print x
```

- ○ 15
- ○ 51 – **correct**
- ○ 150
- ○ An exception is thrown

description: the update method of dictionary will update the values that have the same keys as the newData with newData's values.

Question #80: **What gets printed (with python version 3.X) assuming the user enters the following at the prompt?**
#: foo
47% on 565 times asked

```
a = input("#: ")

print a
```

○ f

[✓ foo – **correct**]

○ Not a number

○ An exception is thrown

description: The input function in python 3.x is the same as the raw_input function in python 2.x.
Therefore foo will be assigned to 'a' and printed.