ADVANCED PYTHON PROGRAMMING

PCEP Preparation

Peter Lo

PCEP-30-02

Certified Entry-Level Python Programmer

What is PCEP?

- **Certified Entry-Level Python Programmer** (PCEP) is the starting point to become certified expert in Python programming.
- This certification shows that the individual is familiar with universal computer programming concepts like data types, containers, functions, conditions, loops, as well as Python programming language syntax, semantics, and the runtime environment.
- Detail information can be obtained from pythoninstitute.org

Python Certification Roadmap

** The PCEP (Entry-Level) certification is **not** a pre-requisite for the PCAP (Associate) or PCPP (Professional) certifications.*

PCEP: Exam Information

Exam Name	PCEP Certified Entry-Level Python Programmer
Exam Code & Version	PCEP-30-02
Duration	45 minutes
Number of Questions	40
Format	• Single-select and multiple-select questions • Drag & drop • Gap fill • Sort
Passing Score	70%
Language	English
Full Exam Price	USD 59

Exam block #1: Computer Programming and Python Fundamentals (18% - 7 exam items)

- PCEP 1.1 Understand fundamental terms and definitions interpreting and the interpreter, compilation and the compiler, lexis, syntax and semantics
- PCEP 1.2 Understand Python's logic and structure keywords, instructions, indenting, comments
- PCEP 1.3 Introduce literals and variables into code and use different numeral systems
- Boolean, integers, floating-point numbers, scientific notation, strings, binary, octal, decimal, and hexadecimal numeral system, variables, naming conventions, implementing PEP-8 recommendations
- PCEP 1.4 Choose operators and data types adequate to the problem numeric operators: ** * / % // + –, string operators: * +, assignments and shortcut operators, operators: unary and binary, priorities and binding, bitwise operators: ~ & ^ | << >>, Boolean operators: not and or, Boolean expressions, relational operators (== != > >= < <=), the accuracy of floating-point numbers, type casting
- PCEP 1.5 Perform Input/Output console operations print(), input() functions, sep= and end= keyword parameters, int() and float() functions

Exam block #2: Control Flow – Conditional Blocks and Loops (29% - 8 exam items)

- PCEP 2.1 Make decisions and branch the flow with the if instruction conditional statements: if, if-else, if-elif, if-elif-else, multiple conditional statements, nesting conditional statements
- PCEP 2.2 Perform different types of iterations the pass instruction, building loops with while, for, range(), and in; iterating through sequences, expanding loops with while-else and for-else, nesting loops and conditional statements, controlling loop execution with break and continue

Exam block #3: Data Collections – Tuples, Dictionaries, Lists, and Strings (25% - 7 exam items)

- PCEP 3.1 Collect and process data using lists constructing vectors, indexing and slicing, the len() function, basic list methods (append(), insert(), index()) and functions (len(), sorted(), etc.), the del instruction; iterating through lists with the for loop, initializing loops; in and not in operators, list comprehensions; copying and cloning, lists in lists: matrices and cubes
- PCEP 3.2 Collect and process data using tuples tuples: indexing, slicing, building, immutability; tuples vs. lists: similarities and differences, lists inside tuples and tuples inside lists
- PCEP 3.3 Collect and process data using dictionaries dictionaries: building, indexing, adding and removing keys; iterating through dictionaries and their keys and values, checking the existence of keys; keys(), items() and values() methods
- PCEP 3.4 Operate with strings constructing strings, indexing, slicing, immutability; escaping using the \ character; quotes and apostrophes inside strings, multi-line strings, basic string functions and methods

Exam block #4: Functions and Exceptions (28% - 8 exam items)

- PCEP 4.1 Decompose the code using functions defining and invoking user-defined functions and generators; the return keyword, returning results, the None keyword, recursion
- PCEP 4.2 Organize interaction between the function and its environment parameters vs. arguments; positional, keyword and mixed argument passing; default parameter values, name scopes, name hiding (shadowing), the global keyword
- PCEP 4.3 Python Built-In Exceptions Hierarchy BaseException, Exception, SystemExit, KeyboardInterrupt, abstractive exceptions, ArithmeticError, LookupError along with IndexError and KeyError; TypeError and ValueError exceptions, the AssertError exception along with the assert keyword
- PCEP 4.4 Basics of Python Exception Handling try-except, try-except Exception, ordering the except branches, propagating exceptions through function boundaries; delegating responsibility for handling exceptions

Sample Question #1

Question #1-1

- Which of these naming conventions for Python identifiers is incorrect?
  A. An identifier can start with a number
  B. An identifier can start with lowercase letter
  C. An identifier can start with underscore(_)
  D. An identifier can start with uppercase letter

Question #1-2

- What refers to assignment of no value in Python?
  A. Zero
  B. Null
  C. Void
  D. None

Question #1-3

- Which of the following words can be used as a variable name? (Choose two.)
  A. for
  B. For
  C. true
  D. True

Question #1-4

- The operation // in python:
  A. Does regular expression
  B. Does simple division
  C. Return the integral part of the quotient
  D. Is not a valid operator

Question #1-5

- Python strings can be glued together using the operator:
  A. .
  B. &
  C. _
  D. +

Question #1-6

- What is the output for the following expression:

```
1 + -2
```

A. -1
B. 1
C. 2
D. The program executed with errors

Question #1-7

- Compute the value of 'x' in the Python expression

```
x = 17 / 2 * 3 + 2
```

A. 4.8
B. 27.5
C. 28
D. 5
E. The program executed with errors

Question #1-8

- What is the output for the following expression:

```
print(14 % 3.5)
```

A. 0
B. 0.0
C. 4
D. 4.0

Question #1-9

- Calculate the value of 'x' in the Python expression

```
x = 19 % 4 + 15 / 2 * 3
```

A. 26
B. 25.5
C. 4.8
D. 5
E. The program executed with errors

Question #1-10

- Determine the value of 'x' in the Python expression

```
x = 17 / 2 % 2 * 3**3
```

A. 14
B. 8.5
C. 13.5
D. 9
E. The program executed with errors

Question #1-11

- What is the output for the following expression:

```
print(10 % 7 % 2)
```

A. -1
B. 0
C. 1
D. The program executed with errors

Question #1-12

- What is the output for the following expression:

```
print(2 ** 2 ** 3)
```

A. 12
B. 64
C. 256
D. The program executed with errors

Question #1-13

- Which of the following literals reflect the value given as 34.23? (Choose two.)
  A. .3423e2
  B. 3423e-2
  C. .3423e-2
  D. 3423e2

Question #1-14

- What is the output for the following code:

```
a = 111111
b = 111_111
print (a + b)
```

A. 1221222
B. 222222
C. 111111111_111
D. The program executed with errors

Question #1-15

- Determine the value of 'x' in the Python expression:

```
a = False
b = 100
x = a + b
```

A. False
B. 100
C. False100
D. The program executed with errors

Question #1-16

- Determine the value of 'x' in the Python expression:

```
a = '1'
b = "1"
print(a + b)
```

A. 2
B. 11
C. 1 1
D. The program executed with errors

Question #1-17

- What is the expected output of the following code?

```
x = '1'
print(x * -5)
```

A. -5
B. 11111
C. It output an empty string
D. The program executed with errors

Question #1-18

- Determine the value of 'x' in the Python expression

```
a = 3E8
b = 300000000
x = a == b
```

A. 3E8
B. 300000000
C. True
D. The program executed with errors

Question #1-19

- What does the following code output?

```
valueOne = 5 ** 2
valueTwo = 5 ** 3
print(valueOne)
print(valueTwo)
```

A. 10 15
B. 25 125
C. The program executed with errors

Question #1-20

- What does the following code output?

```
var1 = 1
var2 = 2
var3 = "3"
print( var1 + var2 + var3 )
```

A. 3
B. 9
C. 126
D. The program executed with errors

Question #1-21

- What is the expected output of the following snippet?

```
s = '*-*'
s = 2*s + s*2
print(s)
```

A. *-**-**-**-*

B. *-**-**-**-**-**-**-*

C. *-*

D. *-**-*

Question #1-22

- What does the following code output?

```
var = "James" * 2 * 3
print(var)
```

A. James
B. JamesJamesJamesJamesJames
C. JamesJamesJamesJamesJamesJames
D. The Program executed with errors

Question #1-23

- What is the output of the following code if you provide input 2 and 3 respectively?

```
num1 = input()
num2 = int( input() )
print( num1 * num2 )
```

A. 6
B. 33
C. 222
D. The program executed with errors

Question #1-24

- What does the following code output?

```
print(bool(0), bool(3.14159), bool(-3), bool(1.0+1j))
```

A. False True True True
B. False True False True
C. True True False True
D. True True False True

Question #1-25

- How many stars (*) does the snippet print?

```
s = '*****'
s = s - s[2]
print[s]
```

A. 2
B. 4
C. 5
D. The program executed with errors

Question #1-26

- Assuming that the following snippet has been successfully executed, which of the equations are True? (Choose two.)

```
a = [1]
b = a
a[0] = 0
```

A. len(a) == len(b)
B. b[0] +1 == a[0]
C. a[0] == b[0]
D. a[0] + 1 == b[0]

Question #1-27

- What does following code output?

```
str = "python"
print ( str[1:3] )
```

A. pyt
B. py
C. yth
D. yt

Sample Question #2

Question #2-1

- What is the purpose of the pass statement?
  A. Jump to loop's else block
  B. Jump to loop condition check
  C. Jump out of the loop
  D. Nothing
  E. None of these

Question #2-2

- What does the following code output?

```
for x in range(10):
    pass
print(x):
```

A. 0
B. 1
C. 9
D. 10

Question #2-3

- What will be the value of the ‘x’ variable when the while loop finishes its execution?

```
x = 0
while x != 0:
    x = x - 1
else:
    x = x + 1
```

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

Question #2-4

- What does the following code output?

```
for x in range(10, 15, 1):
print( x, end=', ')
```

A. 10, 11, 12, 13, 14, 15
B. 10, 11, 12, 13, 14, 15,
C. 10, 11, 12, 13, 14
D. 10, 11, 12, 13, 14,

Question #2-5

- What does the following code provide as an output?

```
for x in range(0.5, 5.5, 0.5):
    print(x)
```

A. [0.5, 1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5, 5.5]
B. [0.5, 1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5]
C. [0.5, 1.5, 2.5, 3.5, 4.5, 5.5]
D. [0.5, 1.5, 2.5, 3.5, 4.5]
E. The Program executed with errors

Question #2-6

- Is it possible to utilize the "else" clause in loops?

```
for x in range(1, 5):
    print(x)
else:
    print("this is else block statement" )
```

A. Yes
B. No

Question #2-7

- How many lines does the following snippet output?

```
for x in range(1, 3):
    print ("*", end="")
else:
    print ("*")
```

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

Question #2-8

- What is the expected output of the following snippet?

```
a = 2
if a > 0:
    a += 1
else:
    a -= 1
print(a)
```

A. 1
B. 2
C. 3
D. The code is erroneous

Question #2-9

- How many stars (*) does the following snippet print?

```
i = 3
while i > 0:
    i -= 1
    print("*")
else:
    print("*")
```

A. 3
B. 4
C. 5
D. The code is erroneous

Question #2-10

- What is the expected output of the following snippet?

```
x = 5
while x > 0:
    x = x // 2
    if x % 2 = 0:
        break:
else:
    x += 1
print(x)
```

A. 3
B. 7
C. 15
D. The program executed with errors

Question #2-10

- What is the expected output of the following code?

```
string = str(1/3)
dummy = ""
for character in string:
    dummy = character + dummy
print(dummy[-1])
```

A. 0
B. 3
C. None
D. The program executed with errors

Question #2-11

- What is the expected behavior of the following code?

```
x = 3 % 1
y = 1 if x > 0 else 0
print(y)
```

A. 1
B. -1
C. 0
D. The program executed with errors

Sample Question #3

Question #3-1

- Assuming that x is a four-element list, is there any difference between these two statements?

```
del x          #The first Line
del x[:]       #The second line
```

A. Yes, there is, the first line empties the list, the second line deletes the list as a whole
B. Yes, there is, the first line deletes the list as a whole, the second line just empties the list
C. Yes, there is, the first line deletes the list as a whole, the second line removes all the elements except the first one
D. No, there is no difference

Question #3-2

- What will the following code do?

```
myList = [0, 1, 2, 3, 4]
myList[0], myList[4] = myList[4], myList[0]
myList[1], myList[3] = myList[3], myList[1]
print(myList)
```

A. This is not allowed in the list
B. It will give an error
C. It will reverse the list
D. It will print [4,1,2,3,0]

Question #3-3

- What is the output for the following code:

```
myList = [7, 12, 5, 4, 27]
print( 27 in myList)
```

A. 27
B. True
C. False
D. The program executed with errors

Question #3-4

- If you run the below code the length of yourList will be same as the length of myList:

```
myList = [ ]
yourList = myList[:]
yourList.append(7)
```

A. True
B. False
C. The program executed with errors

Question #3-5

- What is the expected output of the following snippet?

```
s = 'abc'
for i in len(s):
  s[i] = s[i].upper()
print(s)
```

A. abc
B. ABC
C. 123
D. The program executed with errors

Question #3-6

- Assuming that x is six or more letters long, the following slice is shorter than the original string by:

```
x[1: -2]
```

A. 1 character
B. 2 characters
C. 3 characters
D. 4 characters

Question #3-7

- What is the expected output of the following snippet?

```
x = [1,2,3,4]
x = x[-3:-2]
x = x[-1]
print(x)
```

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

Question #3-8

- How many elements will the list2 list contain after execution of the following snippet?

```
list1 = [False for i in range (1,10)]
list2 = list1[-1:1:-1]
```

A. 0
B. 3
C. 5
D. 7

Question #3-9

- What is the output of the code below?

```
listOne = [20, 40, 60, 80]
listTwo = [20, 40, 60, 80]
print(listOne == listTwo)
print(listOne is listTwo)
```

A. False True
B. True True
C. True False

Question #3-10

- What does the following code output?

```
sampleList = ["Jon", "Kelly", "Jessa"]
sampleList.append(2, "Scott")
print(sampleList)
```

A. [‘Jon’, ‘Kelly’, ‘Jessa’, ‘Scott’]
B. [‘Jon’, ‘Scott’, ‘Kelly’, ‘Jessa’]
C. [‘Jon’, ‘Kelly’, ‘Scott’, ‘Jessa’]
D. The program executed with errors

Question #3-11

- What data type does the following have?

```
x = (1, 'Jhon', 1+3j)
print( type( x[2:3] ) )
```

A. *tuple*
B. *list*
C. *complex*

Question #3-12

- Executing the following snippet will cause the dct:

```
dct = {'pi':3.14}
dct['pi'] = 3.1415
```

A. to hold one key named ‘pi’ linked to 3.1415
B. to hold two keys named ‘pi’ linked to 3.14 and 3.1415 respectively
C. to hold two key named ‘pi’ linked to 3.14 and 3.1415
D. to hold two keys named ‘pi’ linked to 3.1415

Sample Question #4

Question #4-1

- What does the following code output?

```
def calculate (num1, num2=4):
    res = num1 * num2
    print(res)
calculate(5, 6)
```

A. 20
B. 30
C. The program executed with errors

Question #4-2

- If you need a function that does nothing, what would you use instead of XXX?

```
def f( ):
    XXXX
```

A. pass
B. return
C. exit
D. stop

Question #4-3

- Select the valid f( ) invocations: (Choose two)

```
def f(a, b=0):
    return a*b
```

A. f(b=1)
B. f(a=0)
C. f(b=1, 0)
D. f(1)

Question #4-4

- What is the expected output of the following code?

```
str = "abcdef"
def f(s):
    del s[2]
    return s
print(f(str))
```

A. abcef
B. acdef
C. Abdef
D. The program executed with errors

Question #4-5

- What is the expected output of the following snippet?

```
def x():
    return 2
x = 1 + x()
print(x)
```

A. 1
B. 2
C. 3
D. The program executed with errors

Question #4-6

- What is the expected behavior of the following code?

```
def f(x):
    if x % 2 == 1:
        return 0
print( f(1) + f(2) )
```

A. 0
B. 3
C. Print an empty line
D. The program executed with errors

Question #4-7

- Assuming that the following code has been executed successfully, which of the expressions evaluate to True? (Choose two.)

```
def f(x, y):
    nom, denom = x, y
    def g():
        return nom / denom
    return g
a = f(1, 2)
b = f(3, 4)
```

A. b() == 4
B. a != b
C. a is not None
D. a() == 4

Question #4-8

- What is the expected output of the following code?

```
def f(n):
    if n == 1:
        return '1'
    return str(n) + f(n-1)
print(f(2))
```

A. 2
B. 3
C. 12
D. 21
E. The program executed with errors