### **CORE PYTHON**

### **What tool does a programmer use to produce Python source code?**

There are many tools pycharm, spyder, vim, jupyter notebook are few

### **What is necessary to execute a Python program?**

Python compiler/interpreter and python modules before you import in program

### **What is the “official” Python IDE?**

Pythonanywhere

### **What is a statement in a Python program?**

Instructions python interpreter are called statements

### **5. Will the following lines of code print the same thing? Explain why or why not.**

### **x = 6**

### **print(6)**

### **print("6")**

yes, in 1st print statement number 6 and in 2nd print statement string 6 will be printed.

### **6. Will the following lines of code print the same thing? Explain why or why not.**

### **x = 7**

### **print(x)**

### **print("x")**

No, in 1st statement variable x value 7 will be printed, 2nd statement string x will be printed.

### **7. What happens if you attempt to use a variable within a program, and that variable has not been assigned a value?**

We get variable is not defined error

### **8. What is wrong with the following statement that attempts to assign the value ten to variable x? 10 = x**

We get syntax error - SyntaxError: can't assign to literal

### **9. Once a variable has been properly assigned can its value be changed?**

Value of variable can’t be changed. New value will be assigned to another instance of variable

### **10. In Python can you assign more than one variable in a single statement?**

Yes

x,y = 10,20

print(x,y)

### **11. What can you do if a variable name you would like to use is the same as a reserved word?**

When we use reversed words as variables we get syntax error

### **12. How is the value 2.45×10^−5 expressed as a Python literal?**

2.45e-5

### **13. How can you express the literal value 0.0000000000000000000000000449 as a much more compact Python literal?**

0

### **14. How can you express the literal value 56992341200000000000000000000000000000 as a much more compact Python literal?**

**569923412e29**

### **15. Can a Python programmer do anything to ensure that a variable’s value can never be changed after its initial assignment?**

We can create tuples

### **16. Is "i" a string literal or variable?**

String literal

### **17. What is the difference between the following two strings? 'n' and '\n'?**

\n is new line character

### **18. Write a Python program containing exactly one print statement that produces the following output:**

### **A**

### **B**

### **C**

### **D**

### **E**

print('A' '\n' 'B' '\n''C' '\n' 'D' '\n''E')

### **19. Write a Python program that simply emits a beep sound when run.**

import winsound as wd

freq = 1100

dur = 1000

wd.Beep(freq, dur)

### **20. Is the literal 4 a valid Python expression?**

Yes

### **21. Is the variable x a valid Python expression?**

Yes

### **22. Is x + 4 a valid Python expression?**

yes

### **23. What effect does the unary + operator have when applied to a numeric expression?**

No significance

### **24. Sort the following binary operators in order of high to low precedence: +, -, \*, //, /, %, =.**

\*,/,%,+,-,=

### **25. Given the following assignment: x = 2**

### **Indicate what each of the following Python statements would print.**

### **1) print("x") - x**

### **2) print('x') - x**

### **3) print(x) - 2**

### **4) print("x + 1") – x+1**

### **5) print('x' + 1) - error, x must be string**

### **6) print(x + 1) - 3**

### **27. What is printed by the following statement: #print(5/3) – 1.6**

### **28. What symbol signifies the beginning of a comment in Python? - #**

### **29. How do Python comments end?**

Single line comments start with # and comment is consider till end of line. Multiline comments starts and ends with delimiter (“””)

### **30. Which is better, too many comments or too few comments?**

In my opinion its too many or too few, enough comments to understand the program would be better.

### **31. What is the purpose of comments?**

To make other programmers to understand the code

### **32. What circumstances can cause each of the following run-time errors to arise?**

### **1) NameError – whenever local or global name is not available**

### **2) ValueError – whenever operation or some function receives inappropriate value**

x = 5

if x < 10:

raise ValueError('x should not be less than 10!')

### **3) ZeroDivisionError – whenever something divided by 0**

### **3) IndentationError – whenever indentation is not proper**

### **4) OverflowError – whenever arithmetic operation is too large to represent or size is outside required range during list usage**

### **5) SyntaxError – whenever is syntax is wrong. Ex: printd(“hello”)**

### **6) TypeError – whenever wrong type arguemtns passed and incorrect operation or function applied to object etc**

x = 1.5^10000

print(x)

### TypeError: unsupported operand type(s) for ^: 'float' and 'int'

### **Hint: Try some of the following activities in the interpreter or within a Python program:**

### **7) print a variable that has not been assigned**

NameError: name 'variable\_test' is not defined

### **8) convert the string 'two' to an integer**

string = 5

strInt = int(string)

### **9) add an integer to a string**

We will get error. If string is number, we can convert to int format and then add.

### **10) assign to a variable named end-point**

end-point = “hello”

print(end-point)

### **11) experiment adding spaces and tabs at various places in the code of an error-free Python program**

Done

### **12) compute raise a floating-point number to a large power, as in 1.5^10,000**

Overflow error

### **33. What is EOFError ?**

Whenever function reaches end of line in stream without reading data

### **34. Write the shortest way to express each of the following statements.**

### **1) x = x + 1 – x+ =1**

### **2) x = x / 2 – x/= 2**

### **3) x = x – 1 – x- = 2**

### **4) x = x + y - x+ = y**

### **5) x = x - (y + 7) – x- = y-7**

### **6) x = 2\*x – x\* = 2**

### **7) number\_of\_closed\_cases = number\_of\_closed\_cases + 2\*ncc**

**number\_of\_closed\_cases+ = 2\*ncc**

### **35. What is printed by the following code fragment?**

### **x1 = 2**

### **x2 = 2**

### **x1 += 1**

### **x2 -= 1**

### **print(x1)**

### **print(x2)**

3

1

### **Why does the output appear as it does?**

### **36. Consider the following program that attempts to compute the circumference of a circle given the radius entered by the user. Given a circle’s radius, r, the circle’s circumference, C is given by the formula:**

C = 2πr

r = 0

PI = 3.14159

### **#Formula for the area of a circle given its radius**

C = 2*PI*r

### **#Get the radius from the user**

r = float(input("Please enter the circle's radius: "))

### **#Print the circumference**

print("Circumference is", C)

### **The program does not produce the intended result. Why?**

We get syntax error and π is not defined here

### **How can it be repaired so that it works correctly?**

C = 2\*PI\*r and

Input need to be taken from user before computing C

r = 0

PI = 3.14159

r = float(input("Please enter the circle's radius: "))

C = 2\*PI\*r

#Get the radius from the user

print(r)

print("Circumference is", C)

### **37. What possible values can a Boolean expression have? – True and False**

### **38. Where does the term Boolean originate?**

The **word** “**Boolean**” **comes** from the man who invented **Boolean** Logic in the 19th century – George Boole.

### **39. Which is an integer equivalent to True in Python? 1**

### **40. Which is the integer equivalent to False in Python? 0**

### **41. Is the value -16 interpreted as True or False? True**

### **42. Given the following definitions: x, y, z = 3, 5, 7**

### **evaluate the following Boolean expressions:**

### **a) x == 3 - True**

### **b) x < y - True**

### **c) x >= y - Flase**

### **d) x <= y - True**

### **e) x != y – 2 - False**

### **f) x < 10 - True**

### **g) x >= 0 and x < 10 - True**

### **h) x < 0 and x < 10 - False**

### **i) x >= 0 and x < 2 - Flase**

### **j) x < 0 or x < 10 – True**

### **k) x > 0 or x < 10 – True**

### **l) x < 0 or x > 10 - True**

### **43. Express the following Boolean expressions in simpler form; that is, use fewer operators or fewer symbols. x is an integer.**

### **a) not (x == 2) – x != 2**

### **b) x < 2 or x == 2 – x <= 2**

### **c) not (x < y) – x >= y**

### **d) not (x <= y) – x>y**

### **e) x < 10 and x > 20 -**

### **f) x > 10 or x < 20**

### **45. Express the following Boolean expressions in an equivalent form without the not operator. x and y are integers.**

### **a) not (x == y) x != y**

### **b) not (x > y) x !> y**

### **c) not (x < y) x !< y**

### **d) not (x >= y) x < y**

### **e) not (x <= y) x > y**

### **f) not (x != y) y <= x >= y**

### **46. What is the simplest tautology?**

### **47. What is the simplest contradiction?**

### **48. Write a Python program that requests an integer value from the user. If the value is between 1 and 100 inclusive, print ”OK;” otherwise, do not print anything.**

x = int(input("enter value: "))

if 1 <= x <= 100:

print("its in between 1 and 100")

### **49. Write a Python program that requests an integer value from the user. If the value is between 1 and 100 inclusive, print ”OK;” otherwise, print ”Out of range.”**

x = int(input("enter value: "))

if 1 <= x <= 100:

print("its in between 1 and 100")

else:

print("out of range")

### **50. Write a Python program that allows an user to type in an English day of the week (Sunday, Monday, etc.). The program should print the no. of the day as sunday considered day 1.**

day = input("enter day: ")

dic = {'Sunday':1 , 'Monday':2, 'Tuesday':3, 'Wednesday':4, 'Thursday':5, 'Friday':6, 'Saturday':7}

print(dic[day])

### **51.Consider the following Python code fragment:**

### **# i, j, and k are numbers**

### if i < j:

if j < k:

i = j

else:

j = k

else:

if j > k:

j = i

else:

i = k

print("i =", i, " j =", j, " k =", k)

### **What will the code print if the variables i, j, and k have the following values?**

### **a) i is 3, j is 5, and k is 7**

### **b) i is 3, j is 7, and k is 5**

### **c) i is 5, j is 3, and k is 7**

### **d) i is 5, j is 7, and k is 3**

### **e) i is 7, j is 3, and k is 5**

### **52. Consider the following Python program that prints one line of text:**

val = int(input())

if val < 10:

if val != 5:

print("wow ", end='')

else:

val += 1

else:

if val == 17:

val += 10

else:

print("whoa ", end='')

print(val)

### **What will the program print if the user provides the following input?**

### **a) 3 wow 3**

### **b) 21 whoa 21**

### **c) 5 6**

### **d) 17 27**

### **e) -5 wow -5**

### **54. Write a Python program that requests five integer values from the user. It then prints the maximum and minimum values entered. If the user enters the values 3, 2, 5, 0, and 1, the program would indicate that 5 is the maximum and 0 is the minimum. Your program should handle ties properly; for example, if the user enters 2, 4, 2, 3 and 3, the program should report 2 as the minimum and 4 as maximum.**

string = input("enter values separated by ,")

str\_list = string.split(",")

print(max(str\_list))

print(min(str\_list))

### **55. Write a Python program that requests five integer values from the user. It then prints one of two things: if any of the values entered are duplicates, it prints "DUPLICATES"; otherwise, it prints "ALL UNIQUE".**

string = input("enter values")

str\_list = string.split(",")

seen = set()

uniq = [x for x in str\_list if x not in seen and not seen.add(x)]

if len(seen)!= 0:

print("duplicates")

else:

print("unique")

### **56. How many asterisks does the following code fragment print?**

### **57. How many asterisks does the following code fragment print?**

a = 0

while a < 100:

print('\*', end='')

print()

Infinite

### **58. How many asterisks does the following code fragment print?**

a = 0

while a < 100:

b = 0

while b < 55:

print('\*', end='')

b += 1

print()

a += 1

**55**

### **61. How many asterisks does the following code fragment print?**

a = 0

while a < 100:

if a % 5 == 0:

print('\*', end='')

a += 1

print()

20

### **62. How many asterisks does the following code fragment print?**

a = 0

while a < 100:

b = 0

while b < 40:

if (a + b) % 2 == 0:

print('\*', end='')

b += 1

print()

a += 1

0

### **64. How many asterisks does the following code fragment print?**

a = 0

while a < 100:

b = 0

while b < 100:

c = 0

while c < 100:

print('\*', end='')

c += 1

b += 1

a += 1

print()

None

### **65. What is the minimum number of arguments acceptable to the range expression? 1**

### **66. What is the maximum number of arguments acceptable to the range expression? 3**

### **67. Provide the exact sequence of integers specified by each of the following range expressions.**

### **a) range(5) 0 -4**

### **b) range(5, 10) 5- 9**

### **c) range(5, 20, 3) [5, 8,11,14,17]**

### **d) range(20, 5, -1) [20,19,18…7,6]**

### **e) range(20, 5, -3) [20,17,14,11,8]**

### **f) range(10, 5) []**

### **g) range(0) []**

### **h) range(10, 101, 10) [10,20,30,40…100]**

### **i) range(10, -1, -1) [10,9,8…0]**

### **j) range(-3, 4) [-3,-2,-1..3]**

### **k) range(0, 10, 1) [0,1,2..9]**

### **68. What is a shorter way to express range(0, 5, 1)? Range(5)**

### **69. Provide an equivalent Python range expression for each of the following integer sequences.**

### **a) 1,2,3,4,5 range(1,5)**

### **b) 5,4,3,2,1 range(5,0,-1)**

### **c) 5,10,15,20,25,30 range(5,31,5)**

### **d) 30,25,20,15,10,5 range(30,4,-5)**

### **e) −3,−2,−1,0,1,2,3 range(-3,4)**

### **f) 3,2,1,0,−1,−2,−3 range(3,-4,-1)**

### **g) −50,−40,−30,−20,−10 rane(-50,-1,10)**

### **h) Empty sequence range(0)**

### **70. If x is bound to the integer value 2, what integer sequence does range(x, 10\*x, x) represent? [2,4,6..18]**

### **71. If x is bound to the integer value 2 and y is bound to the integer 5, what integer sequence does range(x, x + y) represent? [2,3,4]**

### **72. Is it possible to represent the following sequence with a Python range expression: 1,−1,2,−2,3,−3,4,−4? No**

### **73. How many asterisks does the following code fragment print? 100**

for a in range(100):

print('\*', end='')

print()

### **74. How many asterisks does the following code fragment print? 16**

for a in range(20, 100, 5):

print('\*', end='')

print()

### **75. How many asterisks does the following code fragment print? 50**

for a in range(100, 0, -2):

print('\*', end='')

print()

### **76. How many asterisks does the following code fragment print? 0**

for a in range(1, 1):

print('\*', end='')

print()

### **77. How many asterisks does the following code fragment print? 200**

for a in range(-100, 100):

print('\*', end='')

print()

### **78. How many asterisks does the following code fragment print? 20**

for a in range(-100, 100, 10):

print('\*', end='')

print()

### **79. Rewrite the code in the previous question so it uses a while instead of a for. Your code should behave identically.**

a = 100

while a > -100:

a-=10

print('\*', end='')

### **80. What is printed by the following code fragment? Nothing will be printed**

a = 0

while a > 100:

print(a)

a += 1

print()

### **81. Rewrite the following code fragment using a break statement and eliminating the done variable. Your code should behave identically to this code fragment.**

done = False

n, m = 0, 100

while not done and n != m:

n = int(input())

if n < 0:

done = True

print("n =", n)

n,m = 0,100

while n != m:

n = int(input())

if n == m:

print("n =", n)

break

### **82. Rewrite the following code fragment so it eliminates the continue statement. Your new code’s logic should be simpler than the logic of this fragment.**

x = 5

while x > 0:

y = int(input())

if y == 25:

continue

x -= 1

print('x =', x)

x = 5

while x > 0:

y = int(input())

if y != 25:

x -= 1

print('x =', x)

### **83. What is printed by the following code fragment?**

a = 0

while a < 100:

print(a, end=' ')

a += 1

print()

numbers 0 - 99

### **84. Write a Python program that allows the user to enter exactly twenty floating-point values. The program then prints the sum, average (arithmetic mean), maximum, and minimum of the values entered.**

str\_list = []

a = 0

while a < 20:

in\_value = float(input())

str\_list.append(in\_value)

a += 1

print("sum", sum(str\_list))

print("avg", sum(str\_list)/len(str\_list))

print("max", max(str\_list))

print("min", min(str\_list))

### **85. Write a Python program that allows the user to enter any number of nonnegative floating-point values. The user terminates the input list with any negative value. The program then prints the sum, average (arithmetic mean), maximum, and minimum of the values entered. The terminating negative value is not used in the computations.**

str\_list = []

in\_value = int(input())

while in\_value >= 0:

str\_list.append(in\_value)

in\_value = int(input())

print("sum", sum(str\_list))

print("avg", sum(str\_list)/len(str\_list))

print("max", max(str\_list))

print("min", min(str\_list))

### **86. Write a program : for example, if the user enters 5 the program would print**

### **\***

### **\*\***

### **\*\*\***

### **\*\*\*\***

### **\*\*\*\*\***

### **\*\*\*\***

### **\*\*\***

### **\*\***

### **\***

Executed this in 1st assignment.

### **88. Suppose you need to compute the square root of a number in a Python program. Would it be a good idea to write the code to perform the square root calculation? Why or why not?**

Not a good idea, we already have built in function for that.

### **89. Which of the following values could be produced by the call random.randrange(0, 100) function (circle all that apply)? 4.5 34 -1 100 0 99 It cant be determined, it’s a runtime value**

### **90. Classify each of the following expressions as legal or illegal. Each expression represents a call to a standard Python library function.**

### **a) math.sqrt(4.5) - legal**

### **b) math.sqrt(4.5, 3.1) – illegal**

### **c) random.rand(4) – there is method rand**

### **d) random.seed() – legal**

### **e) random.seed(-1) – legal**

### **91. Write a guessing game program in which the computer chooses at random an integer in the range 1...100. The user’s goal is to guess the number in the least number of tries. For each incorrect guess the user provides, the computer provides feedback whether the user’s number is too high or too low.**

### **92. Is the following a legal Python program?**

def proc(x):

return x + 2

def proc(n):

return 2\*n + 1

def main():

x = proc(5)

main()

Its legal, but always def proc(n): function will be executed.

### **93. Is the following a legal Python program?**

def proc(x):

return x + 2

def main():

x = proc(5)

y = proc(4)

main()

Legal

### **94. Is the following a legal Python program? yes**

def proc(x):

print(x + 2)

def main():

x = proc(5)

main()

### **95. Is the following a legal Python program? - yes**

def proc(x, y):

return 2\*x + y\*y

def main():

print(proc(5, 4))

main()

### **96. Is the following a legal Python program? - yes**

def proc(x):

return 2\*x

def main():

print(proc(5, 4))

main()

### **97. Is the following a legal Python program? - yes**

def proc(x):

print(2\*x\*x)

def main():

proc(5)

main()

### **98. The programmer was expecting the following program to print 200. What does it print instead? Why does it print what it does?**

def proc(x):

x = 2\*x\*x

def main():

num = 10

proc(num)

print(num)

main()

This program will print 10. Because we are not returning value of x from proc function and we are just printing value num in main function.

### **99. Is the following program legal since the variable x is used in two different places (proc and main)? Why or why not?**

def proc(x):

return 2\*x\*x

def main():

x = 10

print(proc(x))

main()

Yes, its legal. Here x is just parameter to receive value from another function

### **100. Complete the following distance function that computes the distance between two geometric points (x1, y1) and (x2, y2):**

### import math as m

### x1,x2,y1,y2 = 4,2,4,2

### res = ((x1 - x2)\*\*2 + (y1-y2)\*\*2)

### print(m.sqrt(res))

### **Test it with several points to convince yourself that is correct.**

### **101. What happens if a caller passes too many parameters to a function?**

Python will throw error like below.

### takes exactly one argument (2 given)

### **102. What happens if a caller passes too few parameters to a function?**

Throws error saying how many parameters required

### **103. What are the rules for naming a function in Python?**

def keyword is used to define function. Function should start with lowercase letter or underscore, can have numbers also, shouldn’t be python keyword

### **104. Consider the following Python code:**

def next\_int1():

cnt = 0

cnt += 1

return cnt

global\_count = 0

def next\_int2():

global\_count += 1

return global\_count

def main():

for i = range(0, 5):

print(next\_int1(), next\_int2())

main()

### **What does the program print?**

It throws error “local variable 'global\_count' referenced before assignment”

### **b) Which of the functions next\_int1 and next\_int2 is the best function for the intended purpose? Why?**

### **c) What is a better name for the function named next\_int1?**

### **d) The next\_int2 function works in this context, but why is it not a good implementation of function that always returns the next largest integer?**

### **105. When is the global statement required?**

Local variables cant be used outside the function. To overcome this global keyword introduced in python3. If we declare any variable with global, that can be used across all functions.

### **106. What does the following Python program print?**

def sum(m=0, n=0, r=0):

return m + n + r

def main():

print(sum())

print(sum(4))

print(sum(4, 5))

print(sum(5, 4))

print(sum(1, 2, 3))

print(sum(2.6, 1.0, 3))

main()

0

4

9

9

6

6.6

m=0, n=0, r=0 – these are default values

### **107. Consider the following function:**

def proc(n):

if n < 1:

return 1

else:

return proc(n/2) + proc(n - 1)

### **Evaluate each of the following expressions:**

### **a) proc(0) - 1**

### **b) proc(1) – 2**

### **c) proc(2) - 4**

### **d) proc(3) – 6**

### **e) proc(5) – 14**

### **f) proc(10) - 60**

### **108. Rewrite the gcd function so that it implements Euclid’s method but uses iteration instead of recursion.**

def gcd\_func(a, b):

r = 1

if a > 0 and b > 0 and type(a) == int and type(b) == int:

while r > 0:

q = a // b

r = a % b

if r > 0:

a = b

b = r

elif r == 0:

print("gcd is ",b)

return b

else:

print("only +ve integers")

### **109. Classify the following functions as pure or impure. x is a global variable.**

a) def f1(m, n):

return 2\*m + 3\*n

b) def f2(n)

return n - 2

c) def f3(n):

return n - x

d) def f4(n):

print(2\*n)

e) def f5(n):

m = int(input())

return m \* n

f) def f6(n):

m = 2\*n

p = 2\*m - 5

return p – n

only f5 is impure function

### **110. Consider the following very simple module, found in the file mymod.py:**

""" Provides the increment function, increment. """

def increment(x):

""" Increments x by 1 and returns the result. """

return x + 1

### **A programmer wishes to use the increment function from the mymod.py module. Indicate which, if any, of the following code snippets would work:**

### **a) import mymod**

### **print(increment(4)) # Supposed to print 5**

### **import from mymod import increment**

### **print(increment(4)) # Supposed to print 5**

### **b) import mymod**

### **print(mymod.increment(4)) # Supposed to print 5**

### **from mymod import increment**

### **print(mymod.increment(4)) # Supposed to print 5**

### **111. Write a generator function named evens that enables the following code:**

### def my\_gen():

### for i in range(12):

### if i % 2 == 0:

### yield i

Caller:

for item in my\_gen():

print(item, end=" " )

### **print 2 4 6 8 10 ; that is, all positive even numbers less than 12**

### **112. Functions as data TODO Consider the following function definition:**

def f():

pass

### **113. Lambda expressions TODO**

### **114. Write a generator function named oscillate that enables the following code:**

### def my\_gen():

### for i in range(-3,5):

### yield i

### yield -i

Caller:

for item in my\_gen():

print(item, end=" " )

### **print -3 3 -2 2 -1 1 0 0 1 -1 2 -2 3 -3 4 -4**

### **115. Local functions TODO**

### **116. Partial application TODO**

### **117. What is the difference between a class and an object?**

Object is instance of class

### **118. What are some other names for the term instance variable?**

Non-static variable

### **119. What is another name for the term method?**

Reference of class

### **120. What symbol associates an object with a method invocation? Dot(.)**

class\_ref = ABC() # object of ABC class

class\_ref.method\_abc() #method invocation

### **121. How does a method differs from a function?**

Method is called by its name associated with an object, it works with data which corresponding class has. Function is block of code and its just called by its name and function doesn’t work with class.

### **122. What method from the string class returns a new string with no leading or trailing whitespace?**

strip()

### **123. What function returns the length of its string argument?**

Len()

### **124. What type of object does the open function return?**

File object

### **125. What does the second parameter of the open function represent?**

Represents the mode of the file while opening(ex: read or write mode)

### **126. Write a program that stores the first 100 integers to a text file named numbers.txt. Each number should appear on a line all by itself.**

f = open(r"C:\training\numbers.txt", "w+")

for i in range(1,101):

f.write(str(i)+ " ")

f.close()

### **127. Complete the following function that reads a collection of integers from a text file named numbers.txt. Each number in the file appears on a line all by itself. The function accepts a single parameter, a string text file name. The function returns the sum of the integers in the file.**

### f = open(r"C:\training\numbers.txt", "r")

### data = f.read()

### lst = []

### lst = data.split(" ")

### lst\_int = [int(i) for i in lst if i != ""]

### print(lst\_int)

### print("sum is", sum(lst\_int)

### f.close()

### **128. Provide the syntactic sugar for each of the following methods of the Fraction class:**

(a) **sub**

(b) **eq**

(c) **neg**

(d) **gt**

### **129. How is using a Turtle object from Python’s Turtle graphics module different from using the free functions; for example, t.penup() versus penup()?**

### **130. Does Python permit a programmer to change one symbol in a string object? If so, how?**

### **131. What would be the consequences if a turtle.Turtle object were immutable?**

### **132. In the context of programming, what is garbage?**

Garbage collection is used for memory management. If any object doesn’t have references to it , that will be considered to delete in next garbage collection run.

### **133. What is garbage collection, and how does it work in Python?**

<https://stackify.com/python-garbage-collection/>

### **134. Consider the following code:**

**a = "ABC"**

**b = a**

**c = b**

**a = "XYZ"**

### **a) At the end of this code’s execution what is the reference count for the string object "ABC"? 2**

### **b) At the end of this code’s execution is b an alias of a? no**

### **c) At the end of this code’s execution is b an alias of c? yes**

### **135. Can a Python list hold a mixture of integers and strings?**

yes

### **136. What happens if you attempt to access an element of a list using a negative index?**

Elements can be accessed from right side using -ve index

### **137. What Python statement produces a list containing the values 45, −3, 16 and 8, in that order?**

### **138. Given the statement**

**(a) What expression represents the very first element of lst? Lst[0]**

**(b) What expression represents the very last element of lst? Lst[-1]**

**(c) What is lst[0]? 10**

**(d) What is lst[3]? 29**

**(e) What is lst[1]? -4**

**(f) What is lst[-1]? 29**

**(g) What is lst[-4]? 10**

**(h) Is the expression lst[3.0] legal or illegal? – Illegal, list index should not be float**

### **140. What function returns the number of elements in a list?**

Len()

### **141. What expression represents the empty list?**

Lst = []

### **142. Given the list**

**lst = [20, 1, -34, 40, -8, 60, 1, 3]**

**evaluate the following expressions:**

**(a) lst**

**(b) lst[0:3]** [20, 1, -34]

**(c) lst[4:8] [-8, 60,1,3]**

**(d) lst[4:33] [-8, 60,1,3]**

**(e) lst[-5:-3] [40,-8]**

**(f) lst[-22:3][20,1,-34]**

**(g) lst[4:]** [-8, 60, 1, 3]

**(h) lst[:][20, 1, -34, 40, -8, 60, 1, 3]**

**(i) lst[:4] [20, 1, -34, 40]**

**(j) lst[1:5]**

**(k) -34 in lst lst[2]**

**(l) -34 not in lst**

**(m) len(lst) 8**

### **143. Write the list represented by each of the following expressions.**

**(a) [8] \* 4 [8,8,8,8]**

**(b) 6 \* [2, 7]** [2, 7, 2, 7, 2, 7, 2, 7, 2, 7, 2, 7]

**(c) [1, 2, 3] + ['a', 'b', 'c', 'd'] [1, 2, 3,'a', 'b', 'c', 'd']**

**(d) 3 \* [1, 2] + [4, 2]** [1, 2, 1, 2, 1, 2, 4, 2]

**(e) 3 \* ([1, 2] + [4, 2])** [1, 2, 4, 2, 1, 2, 4, 2, 1, 2, 4, 2]

### **144. Write the list represented by each of the following list comprehension expressions.**

**(a) [x + 1 for x in [2, 4, 6, 8]] [3,5,7,8]**

**(b) [10\*x for x in range(5, 10)] [50,60,70,80,90]**

**(c) [x for x in range(10, 21) if x % 3 == 0] [12,15,18]**

**(d) [(x, y) for x in range(3) for y in range(4)]** [(0, 0), (0, 1), (0, 2), (0, 3), (1, 0), (1, 1), (1, 2), (1, 3), (2, 0), (2, 1), (2, 2), (2, 3)]

**(e) [(x, y) for x in range(3) for y in range(4) if (x + y) % 2 == 0]** [(0, 0), (0, 2), (1, 1), (1, 3), (2, 0), (2, 2)]

### **145. Provide a list comprehension expression for each of the following lists.**

**(a) [1, 4, 9, 16, 25] [x\*x for x in range(1,6)]**

**(b) [0.25, 0.5, 0.75, 1.0, 1.25. 1.5] [0.25\*x for x in range(1,6)]**

**(c) [('a', 0), ('a', 1), ('a', 2), ('b', 0), ('b', 1), ('b', 2)] [(x,y) for x in [‘a’,’b’] for y in range(3)]**

### **146. If lst is a list, what expression indicates whether or not x is a member of lst?**

Lst.count()

### **147. What does reversed do?**

Returns iterator to access elements in reverse order

### **148. Complete the following function that adds up all the positive values in a list of integers. For example, if list a contains the elements 3,−3,5,2,−1, and 2, the call sum\_positive(a) would evaluate to 12, since 3+5+2+2 = 12. The function returns zero if the list is empty.**

### **def sum\_fun(lst):**

### **a = 0**

### **if len(lst) > 0:**

### **for i in range(len(lst)):**

### **if lst[i] > 0:**

### **a += lst[i]**

### **return a**

### **else:**

### **return 0**

### **149. Complete the following function that counts the even numbers in a list of integers. For example, if list a contains the elements 3,5,4,−1, and 0, the call count\_evens(a) would evaluate to 2, since a contains two even numbers: 4 and 0. The function returns zero if the list is empty. The function does not affect the contents of the list.**

### **def even\_fun(lst):**

### **a = 0**

### **if len(lst) > 0:**

### **for i in range(len(lst)):**

### **if lst[i] % 2 == 0:**

### **a += 1**

### **return a**

### **else:**

### **return 0**

### **150. Write a function named print\_big\_enough that accepts two parameters, a list of numbers and a number. The function should print, in order, all the elements in the list that are at least as large as the second parameter.**

lst = [10,20,0,1,2,3,4,5,6]

lst\_res = [i for i in lst if i > 2 ]

print(lst\_res)

### **151. Write a function named next\_number that accepts a list of integer values. All the elements in the list are unique, and all elements in the list are greater than or equal to one. (The caller must ensure that these conditions are met before passing the list to next\_number.) The next\_number function should return the smallest positive integer not in the list. (Note that 1 is the smallest positive integer.) As examples,**

**• next\_number([5, 3, 1]) would return 2**

**• next\_number([5, 4, 1, 2]) would return 3**

**• next\_number([2, 3]) would return 1**

**• next\_number([]) would return 1**

**lstt = [5,3,53,1]**

**a = max(lstt)**

**b = min(lstt)**

**lst = [i for i in range(1,a)]**

**print(lst)**

**for i in range(len(lst)):**

**if lstt.count(lst[i]) == 0:**

**print("min value not in list is",lst[i])**

**break**

### **152. Write a function named reverse that reorders the contents of a list so they are reversed from their original order. a is a list. Note that your function must physically rearrange the elements within the list, not just print the elements in reverse order.**

lst = ["hello",20,0,1,"hi",3,4,5,6,"how"]

lst\_res = [lst[len(lst)-1-i] for i in range(len(lst))]

print(lst\_res)

### **153. Write a Python program that creates the matrix**

**1 1 1 1 1 1 1 1 1**

**1 1 1 1 1 1 1 1 1**

**1 1 1 1 1 1 1 1 1**

**1 1 1 1 1 1 1 1 1**

**1 1 1 1 1 1 1 1 1**

**1 1 1 1 1 1 1 1 1**

**and assigns it to the variable m. Pretty print m to ensure the contents are correct. Next, reassign m[2][4] to 0, and print m again to ensure your code modified the correct element.**

**w, h = 9, 6;**

**Matrix = [[1 for x in range(w)] for y in range(h)]**

**Matrix[2][4] = 0**

**print(Matrix)**

### **154. Provide five different ways to create the list [1, 2, 3, 4, 5, 6, 7, 8, 9, 10] and assign it to the variable lst.**

num = []

for i in range(1,10):

num.append(i)

print(num)

lst = [i for i in range(10)]

lst

using while loop

### **155. In a square 2D list the number of rows equals the nnumber of columns. Write a function that accepts a square 2D list and returns True if the left to right contents of any row equals the top to bottom contents of any column. If no row matches any column, the function returns False.**

### **156. We can represent a Tic-Tac-Toe board as a 3 × 3 grid in which each position can hold one of the following three strings: "X", "O", or " ". Write a function named check\_winner that accepts a 3 × 3 list as a parameter. If "X" appears in a winning Tic-Tac-Toe pattern, the function should return the string "X". If "O" appears in a winning Tic-Tac-Toe pattern, the function should return the string "O". If no winning pattern exists, the function should return the string " "**

### **157. How are tuples different from lists?**

Main difference is tuples are immutable and lists are mutable

### **158. How do tuples support the indexing operation ([]) differently from lists?**

### **159. Are tuples mutable or immutable?**

Immutable

### **160. Are the elements in tuples ordered or unordered?**

Unoredered

### **161. Rewrite the last assignment statement in the following interactive sequence so that it behaves identically but uses tuple unpacking instead of tuple slicing.**

**a = 1, 2, 3, 4, 5, 6, 7, 8**

**a**

**(1, 2, 3, 4, 5, 6, 7, 8)**

**s = a[2:6]**

**s**

**(3, 4, 5, 6)**

**x,y,\*z,a,b = (1,2,3,4,5,6,7,8)**

**print(z)**

### **162. Consider the tuple tpl defined as tpl = 7, 10, -3, 18, 6, 10**

### **Provide one assignment statement that uses tuple unpacking to assign x to the first element and y to the last element.**

x,\*y,z = (**7, 10, -3, 18, 6, 10**)

### **163. Write a function named zero\_sum that accepts any number of integer arguments. The function should return True if the sum of its arguments is zero; otherwise, it should return False. The call zero\_sum(2, 3, -5), for example, would evaluate to True, since 2 + 3 + −5 = 0. On the other hand, zero\_sum(2, 3, -10, 4) evaluates to False because 2 + 3 + − 10 + 4 = − 1 6= 0. zero\_sum should return True when called with no arguments.**

This can be done

### **164. Why is a dictionary considered an associative container?**

### **165. What statement assigns an empty dictionary to a variable named d?**

D={}

### **166. If d refers to a dictionary, what expression represents the value associated with the key "Fred"?**

D(Fred)

### **168. What happens when an executing program attempts to associate a value with a key that is not present in the dictionary?**

We will get keyError

### **169. Are dictionaries mutable or immutable?**

Mutable

### **170. Given the following dictionary:**

**d = {3:0, 5:1, 10:1, 8:2, 15:4}**

**Indicate what each of the following code fragments will print:**

1. **print(d)**

**{3:0, 5:1, 10:1, 8:2, 15:4}**

**(b) for x in d:**

**print(x)**

3

5

10

8

15

**(c) for x in d.keys():**

**print(x)**

3

5

10

8

15

**(d) for x in d.values():**

**print(x)**

0

1

1

2

4

### **172. Are the elements in dictionaries ordered or unordered?**

Unordered

### **173. Write a graphical, two-player Tic-Tac-Toe game using the tkinter module (see** [**https://en.wikipedia.org/wiki/Tic-tac-toe**](https://en.wikipedia.org/wiki/Tic-tac-toe) **for more information about the game). You can use nine separate variables to track the contents of the game’s squares. You must be able to draw lines and circles in the appropriate locations.**

### **174. Explain why the statement A = {} does not create an empty set.**

It creates empty dictionary

### **175. Provide the Python statement that assigns the variable A to the empty set.**

a = set()

### **176. Are sets mutable or immutable?**

Mutable

### **177. Given the following initialization statements:**

**A = {20, 19, 2, 10, 7}**

**B = {4, 10, 5, 6, 9, 7}**

**C = {10, 19}**

**evaluate the following expressions:**

**(a) A**

**(b) 20 in A - True**

**(c) 20 not in A – False**

**(d) A & B – {10,7}**

**(e) A | B -** {2, 4, 5, 6, 7, 9, 10, 19, 20}

**(f) C < A – True**

**(g) C <= A - True**

**(h) C <= B - True**

**(i) A <= A – True**

**(j) A < A – False**

**(k) len(A) - 5**

**(l) {x + 2 for x in range(10)} - {2,3,4,…11}**

**(m) {x - 2 for x in A} –** {0, 5, 8, 17, 18}

**(n) {x - 2 for x in A if x < 10}** {0, 5}