# Quiz #1 – Fall 2025 Version B - SOLUTION CSCI.UA.0002 - Introduction to Computer Programming

Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

NetID / Email: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**WRITE ALL ANSWERS ON THE ANSWER SHEET. YOU MAY USE THIS QUIZ BOOKLET AS SCRATCH PAPER, BUT NOTHING WRITTEN IN THIS BOOKLET WILL BE GRADED.**

**YOU MUST TURN IN THIS BOOKLET ALONG WITH YOUR ANSWER SHEET AT THE END OF THE QUIZ.**

General notes:

* Please write your answers to each question in the space provided on the answer sheet. Do not write your answers on the test itself!
* When you are finished you must turn in all materials used during the quiz – put them back into your folder and return them to your professor.
* You may detach the “Python Command Index” and “ASCII code table” sheet from the back of the quiz and use it throughout the test. You must return this sheet at the end of the test.
* Scratch paper has been supplied throughout the quiz. None of this scratch work will be graded. The only answers that will be graded are those that you write on the answer sheet.
* Your score will be available in Brightspace once all quizzes have been scored, including make-up quizzes.
* Your professor and the course assistants will be holding extra office hours after scores have been released to answer any questions you may have about how your quiz was graded. Please do not post questions about your grades on Ed, or send them via email.

**Question 1**: Which of the following statements about Python’s if / elif / else control‑flow construct is true?

1. An if block can be followed by multiple elif blocks but only one else block.
2. The else clause must appear immediately after an if block.
3. The else clause can contain a condition to test.
4. All blocks in a chain of if/elif/else are evaluated regardless of which condition matches.
5. elif blocks are evaluated even if a preceding if condition is True.

**Question 2**: Which of the following statements about loops in Python is true?

1. A while loop always executes the loop body at least once, even if the condition is initially False.
2. In a for loop, the target variable can be reassigned inside the loop, and the next iteration will use the new value.
3. A while loop can always be rewritten as a for loop by using the range function, without any change to the loop’s logic.
4. A while loop whose condition never becomes False will automatically terminate after a very large number of iterations to save memory.
5. When using an accumulator variable in a loop (e.g., to sum numbers), the accumulator should be initialized before the loop; if it is initialized inside the loop, it will be reset on each iteration.

**Questions 3-4**: Assume that the following variables are defined in at the top of a Python program:

x = 9

y = 'apple'

z = 3.14

For each of the questions below, determine whether evaluating the expression with the variables above will evaluate to True, False, or raise an Error.

| Question Number | Expression | Options |
| --- | --- | --- |
| 3 | x + z < len(y) or x % 1 == 0 | A. True  B. False  C. Error |
| 4 | len(str.lower(y) \* x) > z and x | A. True  B. False  C. Error |

**Question 5**: Given the following program:

x = 2

y = 4

z = 6

if x < 3:

print("X small")

if y > 3:

print("Y large")

if z < 5:

print("Z small")

elif x == 3:

print("X equal 3")

else:

print("X large")

if y % 2 == 0:

print("Y even")

else:

print("Y odd")

if z == 6:

print("Z perfect")

elif z == 5:

print("Z almost")

else:

print("Z not")

Which of the following options matches the exact output of the program?

| **A.**  X large  Y even  Z not  **B.**  X small  Y large  Y even  Z perfect  **C.**  X small  Y large  Y even | **D.**  X small  Y large  Z perfect  Y even  **E.**  X small  Y even  Z perfect |
| --- | --- |

| ***Questions 6 and 7 refer to the following short program:***  x = 3  y = 2  z = 1  **expr** = x == y+z and y != z+x  **Question 6**: What is the **value** of the shaded expression?   | A. 0  B. 1  C. x == y+z  D. y != z+x  E. 3 and 2 != 4 | AB. True  AC. False  AD. *None*  AE. Error  BC. No choices are correct | | --- | --- |   **Question 7**: What is the **data type** of the shaded expression?   | A. Integer  B. Float  C. String | D. Boolean  E. NoneType  AB. Error | | --- | --- | | ***Questions 10 and 11 refer to the following short program:***  v1 = input("N1: ") # user enters 5  v2 = input("N2: ") # user enters 10  **expr** = format(float(v1 \* 2), ".2f")  **Question 10**: What is the **value** of the shaded expression?   | A. 55  B. 55.0  C. 55.00  D. 10  E. 10.0  AB. 10.00  AC. 1010  AD. 1010.0 | AE. 1010.00  BC. 20  BD. 20.0  BE. 20.00  CD. *None*  CE. Error  DE. No choices are correct | | --- | --- |   **Question 11**: What is the **data type** of the shaded expression?   | A. Integer  B. Float  C. String | D. Boolean  E. NoneType  AB. Error | | --- | --- | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ***Questions 8 and 9 refer to the following short program:***  v1 = int(input("N1: ")) # user enters 7  v2 = int(input("N2: ")) # user enters 3  **expr** = len( str(v1+v2) ) + v1  **Question 8**: What is the **value** of the shaded expression?   | A. 1  B. 3  C. 4  D. 5  E. 6  AB. 7  AC. 8  AD. 9 | AE. 10  BC. 11  BD. 12  BE. 17  CD. *None*  CE. Error  DE. No choices are correct | | --- | --- |   **Question 9**: What is the **data type** of the shaded expression?   | A. Integer  B. Float  C. String | D. Boolean  E. NoneType  AB. Error | | --- | --- | | ***Questions 12 and 13 refer to the following short program:***  v1 = 10  v2 = v1 / 2  v3 = int(v2) / 2  **expr** = v3 + 1  **Question 12**: What is the **value** of the shaded expression?   | A. 2  B. 2.5  C. 3  D. 3.5  E. 4  AB. 4.5  AC. 5 | AD. 5.5  AE. 6  BC. 6.5  BD. *None*  BE. Error  CD. No choices are correct | | --- | --- |   **Question 13**: What is the **data type** of the shaded expression?   | A. Integer  B. Float  C. String | D. Boolean  E. NoneType  AB. Error | | --- | --- | |

**Question 14**: Given the following program:

count = 0

for i in range(5):

for j in range(3):

if j == 1:

break

count += 1

print(count)

How many times is the variable "count" incremented?

A. 0

B. 1

C. 2

D. 3

E. 4

AB. 5

AC. 6

AD. 7

AE. 8

AD. 9

BC. 10

BD. 11

BE. 12

CD. 13

CE. 14

DE. 15

**Questions 15-17** refer to the following three functions:

| def fun1(a, b):  if a > b:  return a - b  else:  return b - a  def fun2(num):  total = 0  for n in range(num):  s = fun1(n, 5)  total += s  print(total)  def fun3(a):  total = fun1(a, 6)  return f"\*{total}\*" | **Question 15:** What will **print** if you call fun1(3, 6)?   | A. 0  B. 1  C. 2  D. 3  E. 4  AB. 5 | AC. 6  AD. Nothing will print  AE. Error  BC. No choices are correct | | --- | --- |   **Question 16:** What will **print** if you call fun2(3)?   | A. 10  B. 11  C. 12  D. 13  E. 14  AB. 15 | AC. 16  AD. Nothing will print  AE. Error  BC. No choices are correct | | --- | --- |   **Question 17**: What is the **return value** of the function call fun3(6)?   | A. -2  B. -1  C. 0  D. 1  E. 2  AB. \*-2\*  AC. \*-1\* | AD. \*0\*  AE. \*1\*  AF. \*2\*  AD. Error  AE. No choices are correct | | --- | --- | |
| --- | --- | --- | --- | --- | --- | --- | --- |

**Question 18**: The program below reads three integer values from the user and then prints a message based on those values.

x = int(input("Enter the first number (x): "))

y = int(input("Enter the second number (y): "))

z = int(input("Enter the third number (z): "))

if x > 10:

if y < 5:

print("Alpha")

else:

print("Beta")

elif x == 10:

if z == 0:

print("Gamma")

else:

print("Delta")

else:

if y == z:

print("Epsilon")

else:

print("Zeta")

Which of the following sets of input values for x, y, z will make the program print the string “Delta”?

1. x=9 y=9 z=9
2. x=8 y=9 z=8
3. x=10 y=7 z=2
4. x=12 y=4 z=0
5. x=10 y=3 z=0

**Question 19**: Given the following program:

x = 0

while x < 4:

x += 1

if x == 2:

continue

print(x)

Which sequence of numbers is printed?

1. 1 3
2. 2 3 4
3. 1 3 4 5
4. 1 2 3 4
5. 1 3 4

**Question 20**: **WRITE YOUR ANSWER TO THIS QUESTION ON THE SECOND PAGE OF THE ANSWER SHEET!**

Refer to the second page of your answer sheet to see an incomplete program designed to roll two virtual dice for the user. It is missing critical pieces of code to make it function as expected, which are represented as empty blanks in the program. **Your task is to fill in these blanks to make the program behave as expected**. Here are some rules:

* You may only write one line of code per blank
* You may not add any additional blanks, or remove any blanks
* You may not change the indentation of any blank

Here's a description of how the program should work:

* The program begins by prompting the user for a positive integer representing the number of sides on a die. For example, if the user enters the number 6 they are indicating that they want to roll a 6 sided die. ***The program only accepts die of size 4 or larger***.
* Next, the program "rolls" two copies of the specified die to result in two random values. For example, on two six sided dice some possible rolls could be (1 and 5), (6 and 2), (3 and 1), etc.
* The program should never roll "doubles" – that is, the same number cannot be rolled on both dice at the same time. For example, with a 6 sided die you could never roll (6 and 6), (5 and 5), (4 and 4), (3 and 3), (2 and 2) or (1 and 1)

Here are a few sample runs of the program. User input is underlined. Note that each run begins with a label to distinguish it from the other runs (e.g. # Run 1) - you do not need to print out this label as part of your program.

| # Run 1  Sides: **4**  You rolled 4 and 3  # Run 2  Sides: **6**  You rolled 2 and 6  # Run 3  Sides: **10**  You rolled 6 and 7  # Run 4  Sides: **0**  Invalid, try again  Sides: **1**  Invalid, try again  Sides: **2**  Invalid, try again  Sides: **4**  You rolled 1 and 3  # Run 5  Sides: **-5**  Invalid, try again  Sides: **-10**  Invalid, try again  Sides: **4**  You rolled 1 and 4 | # Run 6  Sides: **4**  Rolled double 3! Rolling again  Rolled double 2! Rolling again  You rolled 4 and 2  # Run 7  Sides: **12**  Rolled double 12! Rolling again  You rolled 2 and 7  # Run 8  Sides: **20**  Rolled double 14! Rolling again  You rolled 18 and 12  # Run 9  Sides: **4**  Rolled double 4! Rolling again  Rolled double 2! Rolling again  Rolled double 1! Rolling again  Rolled double 1! Rolling again  You rolled 2 and 3 |
| --- | --- |

import random

sides = int(input("Sides: "))

while sides < 4:

print("Invalid, try again")

sides = int(input("Sides: "))

die1 = random.randint(1, sides)

die2 = random.randint(1, sides)

while die1 == die2:

print(f"Rolled double {die1}! Rolling again")

die1 = random.randint(1, sides)

die2 = random.randint(1, sides)

print("You rolled", die1, "and", die2)

**Question 21**: **WRITE YOUR ANSWER TO THIS QUESTION ON THE THIRD PAGE OF THE ANSWER SHEET!**

Write a **FUNCTION** named pattern\_generator based on the following IPO notation:

Function: pattern\_generator

Input: A pattern type (string) - must be "triangle"

The size of the pattern (integer) - must be an integer >= 3

Processing: This function will print out a pattern based on the supplied pattern type

and size. If an invalid pattern type or size is detected, the function

can simply end and generate no output.

All patterns will print using a series of "#" symbols.

"triangle" patterns will generate a right-facing triangle. For example,

calling pattern\_generator("triangle", 3) would print a pattern with a max

width of the triangle being 3 (see below for sample output)

Output: Prints the desired pattern but returns nothing.

Ensure that your function operates **exactly** as specified in the IPO notation and in the sample calls below. Do not add any additional features that are not described here. **Write your answer on the answer sheet in the space designated for Question 21.** You can always assume the function will be called with three integers.

You are only writing a function for this question. Do not worry about writing a main program. Sample calls:

| pattern\_generator("", 0)  pattern\_generator("apple", 5)  pattern\_generator("square", -5)  pattern\_generator("triangle", 4)  pattern\_generator("triangle", 3) | # no output  # no output  # no output  #  ##  ###  ####  ###  ##  #  #  ##  ###  ##  # |
| --- | --- |

For extra credit you can extend your function so that it can generate a second pattern type, called "square" which produces a hollow square of the desired size. For example:

| pattern\_generator("square", 4)  pattern\_generator("square", 3) | ####  # #  # #  ####  ###  # #  ### |
| --- | --- |

def pattern\_generator(type\_of\_pattern, size):

if size < 3:

return

if type\_of\_pattern == "triangle":

for c in range(1, size):

print("#" \* c)

for c in range(size, 0, -1):

print("#" \* c)

**Python Command and Function Index**

| and  break  continue  def  elif  else  float  for  format  if  in | import  input  int  len  not  or  print  return  str  while | random.randint  str.lower  str.upper  time.time |
| --- | --- | --- |

**ASCII Code Table**

