PYTHON TEST - 1.13 (MORE ON LOOPS)

Total points 50/50

More on Loops

STUDENT NAME *

VIVA

✓ 1. The else block of a loop executes when: * 1/1

- a) The loop executes at least once
- b) The loop is terminated with a break
- c) The loop finishes normally without break ✓
- d) The loop condition is False at the beginning

✓ 2. . Which of the following loops can have an else clause? * 1/1

- ○ a) for loop
- ○ b) while loop
- ◉ c) Both for and while ✓
- ○ d) Neither

✓ 3.What will be the output? * 1/1

```
for i in range(3):
    print(i)
else:
    print("Done")
```

- ○ a) 0 1 2
- ◉ b) 0 1 2 Done ✓
- ○ c) Done
- ○ d) Error

✓ 4. When will the else block NOT execute in a loop? * 1/1

- ○ a) When loop finishes normally
- ○ b) When loop has no iterations
- ◉ c) When loop ends with break ✓
- ○ d) Always executes

✓ 5. Output? * 1/1

```
n = 0
while n < 3:
    print(n)
    n += 1
else:
    print("Finished")
```

- ○ a) 0 1 2
- ◉ b) 0 1 2 Finished ✓
- ○ c) Finished
- ○ d) Infinite loop

✓ 6. Does else run if loop body never executes but no break is used? * 1/1

- ◉ a) Yes ✓
- ○ b) No
- ○ c) Only in for
- ○ d) Only in while

✓ 7.What will be output? * 1/1

```
for i in range(5):
    if i == 2:
        break
else:
    print("Completed")
```

- ○ a) Completed
- ○ b) 0 1 Completed
- ○ c) 0 1
- ◉ d) 0 1 2 ✓

✓ 8. Which statement is correct about loop-else in Python? * 1/1

- ○ a) It acts like finally in exception handling
- ○ b) It runs only if break is encountered
- ◉ c) It runs if loop terminates normally ✓
- ○ d) It cannot be used in while loops

✓ 9.Output? * 1/1

```
for i in []:
    print(i)
else:
    print("Empty loop")
```

- ○ a) No output
- ○ b) Error
- ◉ c) Empty loop ✓
- ○ d) None

✓ 10. In which case will while...else not execute else? * 1/1

- ○ a) Loop runs zero times
- ○ b) Loop condition false initially
- ◉ c) Loop ends with break ✓
- ○ d) All of the above

✓ 11. The else with loop is most similar to: * 1/1

- ○ a) catch
- ○ b) finally
- ○ c) default
- ◉ d) try without exception ✓

✓ 12. Can else be used without a loop? * 1/1

- a) Yes
- b) No ✓

✓ 13. Loop-else is mainly useful with: * 1/1

- a) Searching problems ✓
- b) Sorting problems
- c) File handling
- d) Functions

✓ 14.Output? * 1/1

```
for i in range(3):
    print(i)
else:
    print("Loop ended")
```

- a) 0 1 2
- b) 0 1 2 Loop ended ✓
- c) Loop ended
- d) Error

✓ 15. Can we combine else with continue inside a loop? * 1/1

- (●) a) Yes ✓
- ( ) b) No

✓ 16. Loop-else helps differentiate between: * 1/1

- (●) a) Loop ending normally vs abnormal exit ✓
- ( ) b) Infinite loop vs finite loop
- ( ) c) For vs while loop
- ( ) d) None

✓ 17. Which keyword affects loop-else execution? * 1/1

- ( ) a) continue
- ( ) b) pass
- (●) c) break ✓
- ( ) d) return

✓ 18. In a while loop, else executes when: * 1/1

- a) Break occurs
- b) Condition becomes False ✓
- c) Condition becomes True
- d) Never

✓ 19.Output? * 1/1

```
for i in range(2):
    print("Hi")
else:
    print("Bye")
```

- a) Hi Hi
- b) Hi Hi Bye ✓
- c) Bye
- d) Error

✓ 20. Which is correct? * 1/1

- a) else is optional in loops ✓
- b) else is mandatory in loops
- c) else can only be used with while
- d) None

✓ 21.Output? * 1/1

```
while False:
    print("Running")
else:
    print("Not running")
```

- ○ a) Running
- ◉ b) Not running ✓
- ○ c) No output
- ○ d) Error

✓ 22. Can loop-else be nested? * 1/1

- ◉ a) Yes ✓
- ○ b) No

✓ 23.Output? * 1/1

```
for i in range(3):
    if i == 1:
        break
else:
    print("Loop finished")
```

- ○ a) Loop finished
- ○ b) 0
- ○ c) 0 1 Loop finished
- ◉ d) 0 ✓

✓ 24.Output? * 1/1

```
for i in range(2):
    pass
else:
print("Done")
```

- ◉ a) Done ✓
- ○ b) No output
- ○ c) Error
- ○ d) None

✓ 25. Loop-else is rarely used, but useful when: * 1/1

- (●) a) Searching elements ✓
- ( ) b) Reversing lists
- ( ) c) Declaring variables
- ( ) d) Printing statements

✓ 26. What is a nested loop? * 1/1

- (●) a) Loop inside another loop ✓
- ( ) b) Loop with else
- ( ) c) Loop with break
- ( ) d) None

✓ 27. Which is valid syntax for nested loops? * 1/1

- ( ) a) for..while
- ( ) b) while..for
- ( ) c) for..for
- (●) d) All of the above ✓

✓ 28.Output? * 1/1

```
for i in range(2):
    for j in range(2):
        print(i, j)
```

- (●) a) 0 0 0 1 1 0 1 1 ✓
- ( ) b) 0 1 1 0
- ( ) c) 0 0 1 1
- ( ) d) Error

✓ 29. In nested loops, the inner loop executes: * 1/1

- ( ) a) Once
- ( ) b) Depends on break
- (●) c) Fully for each outer loop iteration ✓
- ( ) d) Randomly

✓ 30.Output? * 1/1

```
count = 0
for i in range(3):
    for j in range(2):
        count += 1
print(count)
```

- a) 3
- b) 5
- c) 6 ✓
- d) Error

✓ 31. Nested loop complexity multiplies when: * 1/1

- a) Both loops run independently ✓
- b) Loops depend on condition
- c) Loop else is used
- d) None

✓ 32.Output? * 1/1

```
for i in range(2):
    for j in range(3):
        print("*", end="")
    print()
```

- ○ a) ** (new Line) **
- ◉ b) *** (new Line) *** ✓
- ○ c) **
- ○ d) Error

✓ 33.What is the time complexity of: * 1/1

```
for i in range(n):
    for j in range(n):
        print(i, j)
```

- ○ a) O(n)
- ◉ b) O(n^2) ✓
- ○ c) O(log n)
- ○ d) O(1)

✓ 34. Nested loops are often used in: * 1/1

- (●) a) Matrices ✓
- ( ) b) Variables
- ( ) c) Functions
- ( ) d) Strings

✓ 35.Output? * 1/1

```
for i in range(2):
    for j in range(2):
        print(i+j, end=" ")
```

- (●) a) 0 1 1 2 ✓
- ( ) b) 0 1 2 3
- ( ) c) 0 0 1 1
- ( ) d) 2 2

✓ 36. Maximum nesting allowed in Python is: * 1/1

- ( ) a) 10
- (●) b) 1000 (based on recursion limit) ✓
- ( ) c) Unlimited
- ( ) d) None

✓ 37. In nested loops, which loop finishes first? * 1/1

- a) Outer
- b) Inner ✓
- c) Both simultaneously
- d) None

✓ 38.Output? * 1/1

```
for i in range(3):
    for j in range(i):
        print(j, end=" ")
```

- a) 0 1 2
- b) 0 0 1 ✓
- c) 1 2 3
- d) None

✓ 39. Nested loops can also contain: * 1/1

- a) Functions
- b) Conditionals
- c) Break/continue
- d) All ✓

✓ 40.Output? * 1/1

```
for i in range(2):
    for j in range(2):
        for k in range(2):
            print(i, j, k)
```

- ○ a) 4 outputs
- ○ b) 6 outputs
- ◉ c) 8 outputs ✓
- ○ d) Error

✓ 41. Nested loops are mostly used in: * 1/1

- ◉ a) Pattern printing ✓
- ○ b) Function calls
- ○ c) Imports
- ○ d) Return statements

✓ 42.Output? * 1/1

```
for i in range(2):
    for j in range(2):
        if j == 1:
            break
    print(i, j)
```

- a) 0 0 1 0
- b) 0 0 1 0 ✓
- c) 0 1 1 1
- d) None

✓ 43. Which loop is more efficient to nest with lists? * 1/1

- a) for ✓
- b) while
- c) Both
- d) None

✓ 44.Output? * 1/1

```
for i in range(2):
    for j in range(3):
        if i == j:
            print(i, j)
```

- (●) a) 0 0 1 1 ✓
- ( ) b) 0 1 2
- ( ) c) 0 1
- ( ) d) None

✓ 45. Nested loops can also be combined with: * 1/1

- (●) a) Else ✓
- ( ) b) Except
- ( ) c) Finally
- ( ) d) Raise

✓ 46.Output? * 1/1

```
for i in range(2):
    for j in range(2):
        print("Hello")
    else:
        print("Inner done")
```

- ○ a) Hello Hello Hello Hello Inner done
- ◉ b) Hello Hello Inner done Hello Hello Inner done ✓
- ○ c) Error
- ○ d) None

✓ 47. Nested loops are often avoided in big programs due to: * 1/1

- ◉ a) High time complexity ✓
- ○ b) Memory leak
- ○ c) Syntax issues
- ○ d) Compiler error

✓ 48.Output? *

1/1

```
for i in range(3):
    for j in range(2):
        print("*", end="")
    print()
```

- (●) a) ** ** ** ✓
- ( ) b) * * *
- ( ) c) ***
- ( ) d) Error

✓ 49. Nested loops can iterate over: *

1/1

- ( ) a) Lists
- ( ) b) Strings
- ( ) c) Dictionaries
- (●) d) All ✓

✓ 50.Output? *

1/1

```
for i in range(2):
    for j in range(2):
        print(i*j, end=" ")
```

- a) 0 0 0 0
- b) 0 0 0 1 ✓
- c) 0 1 2 3
- d) None