Programming in Python
Midterm Exam Solutions

Karlo Knežević
karloknezevic.github.io

November 23, 2021

Info:
37 points, 92.5 min.

Solutions

Learning outcome 1

Question 1

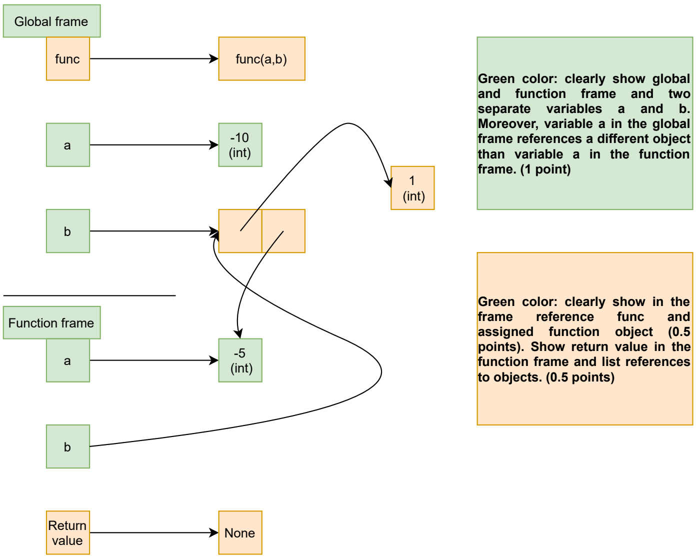
Answer 1 2/0 points

(a) Keyword *is* compares objects **by reference** (0.5 points). Operator == compares objects' values (0.5 points).

(b) Python uses a **memory pool** and *a* and *b* **reference the same object** (1 point).

Question 2

Answer 2 0/2 points


Global frame
func
func(a,b)
a
-10
(int)
b
1
(int)
Function frame
a
-5
(int)
b
Return
value
None
Green color: clearly show global and function frame and two separate variables a and b. Moreover, variable a in the global frame references a different object than variable a in the function frame. (1 point)
Green color: clearly show in the frame reference func and assigned function object (0.5 points). Show return value in the function frame and list references to objects. (0.5 points)

Question 3

Answer 3 1/1 points

```python
l = [(chr(e), chr(e+1).upper()) for e in range(ord("a"), ord("z"), 2)]

print(l)
```

Comment:

- 1 point is given if list comprehension is properly used and if for loop is inside of the list
- 1 point is given for a correct solution

Question 4

Answer 4 4/0 points

```
import random

while True:
    n = int(input("Enter␣n:"))

    if n > 0:
        break

    print("n␣must␣be␣positive")

l = [random.randint(-10, 10) for i in range(n)]

print("Unique␣elements".center(80, "-"))
print(f"{len(set(l)):>80d}")

d = {}
for i in l:
    d[i] = d.get(i, 0) + 1
print("Statistics".center(80, "-"))
for k in sorted(d):
    print(f"{k:>3d}{d[k]:>77d}")
max_ = max(d.values())
print("Most␣frequent␣elements".center(80, "-"))
for k in sorted(d):
    if d[k] == max_:
        v = f"{k}({d[k]})"
        print(f"{v:^80s}")
```

Comment:

- 0.5 points for *while-True* loop for validation
- 0.5 points for a list generation (list comprehension or any initialization loop)
- 1 point: (0.5) for centered title and (0.5) for right alignment
- 1 point: (0.5) for left alignment and (0.5) for right alignment
- 1 point: (0.5) for centered value and frequency and (0.5) if handled multiple same frequencies

Learning outcome 2

Question 5

Answer 5 1/0 points

In function annotations -> is used in function declaration to announce the **returning value type**.

Question 6

Answer 6 3/0 points

```
def calculate(operands, operators):
    if len(operands) == 1:
        return operands[0]
    else:
        result = operands[0]
        for i in range(1, len(operators) + 1):
            if operators[i - 1] == '+':
                result += operands[i]
            elif operators[i - 1] == '-':
                result -= operands[i]
            elif operators[i - 1] == '*':
                result *= operands[i]
            elif operators[i - 1] == '/':
                if operands[i] == 0:
                    return
                result /= operands[i]
        return result

#prints the result of the calculation
# 1 + 2 - 3 * 4 + 5 = -5
print(calculate([1, 2, 5, 4, 3], ['+', '-', '*', '+']))
#prints the result of the calculation
# 1 + 2 - 3 * 4 + 5 = 0
print(calculate([1, 2, 3, 4, 5], ['+', '-', '*', '/']))
```

Comment:

- 1 point: function declaration and return keyword
- 1 point: comparing operators and arithmetic operations
- 1 point: (0.5) if handled division by zero and (0.5) solution correct

Question 7

Answer 7 3/0 points

```
def my_letters(start, end = "z", step = 1):
    counter = 1
    while ord(start) <= ord(end):
        yield (counter, start)
        start = chr(ord(start) + step)
        counter += 1

#(1, ’a’) (2, ’f’) (3, ’k’) (4, ’p’) (5, ’u’) (6, ’z’)
for letter in my_letters("a", "z", 5):
    print(letter, end = "␣")
```

Comment:

- 1 point: (0.5) function declaration and (0.5) default value of step
- 1 point: using *yield* keyword and returning tuple
- 1 point: (0.5) printing in the same line and (0.5) solution correct

Question 8

Answer 8 0/3 points

```
def fibonacci(n:int) -> int:
    """
    Returns the nth fibonacci number
    """
    if n <= 1:
        return n
    return fibonacci(n - 1) + fibonacci(n - 2)
```

Comment:

- 1 point: recursive call
- 1 point: (0.5) stopping criterion and (0.5) function annotations
- 1 point: Docstring

Learning outcome 3

Question 9

Answer 9 1/1 points

(1 point) Package **is a directory** that contains Python modules. (1 point) **__init__.py** is the first executed module after importing the package (is existing).

Question 10

Answer 10 2/1 points

```
_b = 10

def adder(a, b):
    return a + b

if __name__ == ’__main__’:
    print(adder(5, _b))
```

```
from lib import *

if __name__ == ’__main__’:
    print(adder(5, 10))
```

Comment:

- 1 point: hidden variable name must begin with underscore
- 1 point: in lib.py must be `if __name__ == "__main__"` construct
- 1 point: import lib.py and call the `adder` function

Learning outcome 5

Question 11

Answer 11 1/0 points

(1 point) The benefit of using with block when working with files is **no need to explicitly close** the opened file.

Answer 12 5/0 points

```
with open("a.txt", "a") as a:
    with open("b.txt") as b:
        for line in b:
            if not line.endswith("\n"):
                line += "\n"
            a.write(line)
```

Comment:

- 1 point: open and read one file
- 1 point: open a file with mode "a"
- 1 point: handle situation if line does not end with a new line character
- 2 points: close both files (by default with `with` block)

Answer 13 3/0 points

```python
import random
lines = open('main.py').read().splitlines()

if len(lines) < 1:
    print("Empty␣file!")
else:
    myline = lines[random.randint(0, len(lines)-1)]
    print(myline)
```

Comment:

- 1 point: read file into memory
- 1 point: check if file empty
- 1 point: print a random line

Answer 14 1/2 points

```python
import random
import json

l = [random.randint(-100, 100) for _ in range(20)]

with open("numbers.txt", "w") as f:
    json.dump(l, f)

with open("numbers.txt") as f:
    print(f.read())
```

Comment:

- 1 point: generate a list of random numbers
- 1 point: import *json* module and use *dump* method to serialize data
- 1 point: print raw data to the screen