(1-3) Given the following Python object: 99.9

1) <class 'float'> refers to the object
   a) type           b) value       c) memory location d) identifier e) keyword

2) 99.9 refers to the object
   a) type           b) value       c) memory location d) identifier e) keyword

3) 4298470336 refers to the object
   a) type           b) value       c) memory location d) identifier e) keyword

(4) Which expanded description of binary 1101 is correct?
   a) $1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$
   b) $1 \times (2^3 + 1) \times (2^2 + 0) \times (2^1 + 1) \times 2^0$
   c) $1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$
   d) $1 \times 2^3 + 1 \times 2^1 + 0 \times 2^2 + 1 \times 2^0$
   e) $1 \times 2^1 + 1 \times 2^0 + 0 \times 2^1 + 1 \times 2^0$

(5-9) For each Python expression, indicate the type of error it would generate (if any):

5) >>> '1, 2, 3, 4, ' + 5
   a) SyntaxError   b) NameError   c) TypeError      d) IndexError e) no error

6) >>> len(1000)
   a) SyntaxError   b) NameError   c) TypeError      d) IndexError e) no error

7) >>> len(mystring)
   a) SyntaxError   b) NameError   c) TypeError      d) IndexError e) no error

8) >>> 4 $ 3
   a) SyntaxError   b) NameError   c) TypeError      d) IndexError e) no error

9) >>> 'forty'[5]
   a) SyntaxError   b) NameError   c) TypeError      d) IndexError e) no error
def fizzbuzz(n):
    '''(int) -> None

    Play fizzbuzz up to n.
    Results are printed during play;
    None value is returned
    '''
    for i in range(1, n+1):
        m3 = (i % 3) == 0
        m5 = (i % 5) == 0

        if m3 and m5:
            print('fizzbuzz')
        elif m3:
            print('fizz')
        elif m5:
            print('buzz')
        else:
            print(i)

    print('Game over!')

    return None

(10) for is a Python

a) keyword    b) identifier    c) primitive element    d) namespace    e) library module

(11) i is a Python

a) keyword    b) identifier    c) primitive element    d) namespace    e) library module

(12) m3 is a Python

a) keyword    b) identifier    c) primitive element    d) namespace    e) library module

(13) 3 is a Python

a) keyword    b) identifier    c) primitive element    d) namespace    e) library module

(14) if is a Python

a) keyword    b) identifier    c) primitive element    d) namespace    e) library module

(15) 'fizzbuzz' is a Python

a) keyword    b) identifier    c) primitive element    d) namespace    e) library module
The following Python code does not work as indicated in the (correct) function docstrings:

```python
1 def est_tax(income, exemptions):
    '''(number, int) -> float

    Calculates and prints an estimate for federal income tax
    for the given income and exemptions. Standard exemption is
    $4,150 per exemption, standard deduction is $6,500, and the
tax rate is a flat 20%.

    CALLS: taxable

    >>> est_tax(20000, 1)
    Estimated tax is: 1870.0
    '''
2    STD_EXEMPT = 4150
3    STD_DEDUCT = 6500
4    TAX_RATE = .20
5    taxable_income = taxable(income, exemptions, STD_EXEMPT, STD_DEDUCT)
6    estimated_tax = taxable_income * TAX_RATE
7    print(f'Estimated tax is: {estimated_tax}')
8    return None

9 def taxable(income, exemptions, STD_EXEMPT, STD_DEDUCT):
    '''(number, int, int, int) -> float

    Adjust gross income (income) to taxable income
    by applying standard deduction and exemptions.
    Returns taxable income.

    CALLED BY: est_tax

    >>> taxable(20000, 1, 4150, 6500)
    9350
    '''
10   taxable_income = income - STD_DEDUCT
11   exempt_adjust = STD_EXEMPT * exemptions
12   taxable_income = taxable_income - exempt_adjust
13   print(taxable_income)
14   return None
```

(16) What type of error is generated when ```>>> est_tax(20000, 1)``` is executed?

a) syntax  

b) runtime  

c) logic/semantic  

d) None

(17) Which lines of code need to be changed for the functions to work correctly?

a) 1, 7, 8  

b) 1, 8  

c) 2, 3, 4  

d) 9, 13, 14  

e) 13, 14
(18) When function taxable is called from est_tax, which of the following happens:

a) an activation record for function taxable is removed from the call stack

b) STD_EXEMPT and STD_DEDUCT are added to the global namespace

c) the estimated tax is printed

d) all of these
e) none of these

(19-24) (.5 pts. each) Given the following Python code:

```python
def twice(x):
    '''test function'''
    y = 2
    result = x * 2
    return result

>>> y = 5
>>> twice(y)
-1
>>> y
2
>>> x
3
```

(19) x is defined in a ?? namespace:

a) local
b) global
c) built-in
d) both local and global
e) none of these

(20) y is defined in a ?? namespace:

a) local
b) global
c) built-in
d) both local and global
e) none of these

(21) twice is defined in a ?? namespace:

a) local
b) global
c) built-in
d) both local and global
e) none of these

(22) The value reported at ??-1 is:

a) 4  b) 5  c) 10  d) 25  e) None

(23) The value reported at ??-2 is:

a) 2  b) 5  c) 10  d) None  e) error

(24) The value reported at ??-3 is:

a) 2  b) 5  c) 10  d) None  e) error
(25-30) (.5 pts. each) Given the following Python code:

```python
def thrice(x, y):
    '''test function'''
    z = 3
    print(x, y, z)
    result = z * x * y
    return result

>>> x = 5
>>> y = 10
>>> answer = thrice(y, x)
>>> answer
```

(25) What will be printed at line 4 when function `thrice` executes?

a) 3, 5, 10  
   b) 5, 10, 3  
   c) 10, 5, 3  
   d) None  
   e) error

(26) The value reported at ??-1 is

a) 3  
   b) 5  
   c) 10  
   d) 150  
   e) error

(27) The value reported at ??-2 is

a) 3  
   b) 5  
   c) 10  
   d) 150  
   e) error

(28) The value reported at ??-3 is

a) 3  
   b) 5  
   c) 10  
   d) 150  
   e) error

(29) The value reported at ??-4 is

a) 3  
   b) 5  
   c) 10  
   d) 150  
   e) error

(30) The value reported at ??-5 is

a) 3  
   b) 5  
   c) 10  
   d) 150  
   e) error
Given the following Python code:

```python
from math import pi

def circle_area(rad):
    '''(num) -> float
    return the area of a circle with radius rad
    >>> circle_area(1)
    3.1
    >>> circle_area(2)
    12.6
    '''
    area = pi * rad ** 2
    area = round(area, 1)
    return area

def pizza_calculator(diameter, cost):
    '''
    (int, num) -> float
    Calculates and returns the cost per square inch
    of pizza for a pizza of given diameter and cost.
    >>> pizza_calculator(14, 18)
    0.117
    >>> pizza_calculator(14, 20.25)
    0.132
    '''
    r = diameter / 2
    area = pi * r ** 2
    cost_per_inch = cost / area
    cost_per_inch = round(cost_per_inch, 3)
    return cost_per_inch
```

31) Which of the following would you expect to see in the `__main__` namespace after `from math import pi` is executed?

a) import  b) math  c) pi  d) math.pi  e) math.sqrt

32) Which line of code below should replace lines 13 and 14 in `pizza_calculator` so that the pizza area calculation is moved to `circle_area`?

a) `area = circle_area(diameter / 2)`
b) `area = circle_area(rad)`
c) `circle_area(diameter / 2)`
d) `circle_area(rad)`
(33-38) (.5 pts. each) After the following Python code is executed:

```python
>>> x = 99.9
>>> y = x
>>> z = 99.9
```

What is the result of executing the following Python code?

(33) >>> x == y

a) True  
b) False  
c) 99.9  
d) <class 'float'>

(34) >>> id(x) == id(y)

a) True  
b) False  
c) 99.9  
d) <class 'float'>

(35) >>> x == z

a) True  
b) False  
c) 99.9  
d) <class 'float'>

(36) >>> id(x) == id(z)

a) True  
b) False  
c) 99.9  
d) <class 'float'>

In the same interactive Shell session, after the following Python code is executed:

```python
>>> x = 100
```

What is the result of executing the following Python code?

(37) >>> id(x) == id(y)

a) True  
b) False  
c) 99.9  
d) <class 'int'>

(38) >>> id(y) == id(z)

a) True  
b) False  
c) 99.9  
d) <class 'int'>
(39-44) (.5 pts. each) After the following Python code is executed:

```python
>>> x = [1, 2, 3]
>>> y = x
>>> z = [1, 2, 3]
```

What is the result of executing the following Python code?

(39) >>> x == y  

a) True    

b) False   

c) [1, 2, 3]  

d) <class 'list'>

(40) >>> id(x) == id(y)

a) True    

b) False   

c) [1, 2, 3]  

d) <class 'list'>

(41) >>> x == z

a) True    

b) False   

c) [1, 2, 3]  

d) <class 'list'>

(42) >>> id(x) == id(z)

a) True    

b) False   

c) [1, 2, 3]  

d) <class 'list'>

In the same interactive Shell session, after the following Python code is executed:

```python
>>> x[2] = 99
```

What is the result of executing the following Python code?

(43) >>> x == y

a) True    

b) False   

c) [1, 2, 3]  

d) <class 'list'>

(44) >>> id(x) == id(y)

a) True    

b) False   

c) [1, 2, 3]  

d) <class 'list'>
(45-48) After the following Python code is executed:

```python
>>> mylist = [1, False, 'hi']
>>> mylist[1] = 99
```

(45) The value of mylist is:

a) [99, False, 'hi']  b) [1, 99, 'hi']  c) [1, False, 99]
d) 99  e) None

In the same interactive Shell session, after the following Python code is executed:

```python
>>> mylist = mylist.reverse()
```

(46) The value of mylist is:

a) ['hi', False, 99]  b) ['hi', 99, 1]  c) [99, False, 1]
d) 99  e) None

In the same interactive Shell session, after the following Python code is executed:

```python
>>> mylist = [10, 12, 2]
>>> yourlist = mylist
>>> mylist.reverse()
```

(47) The value of mylist is:

a) [10, 12, 2]  b) [2, 12, 10]  c) [10, 12, 2, 2, 12, 10]
d) []  e) None

(48) The value of yourlist is:

a) [10, 12, 2]  b) [2, 12, 10]  c) [10, 12, 2, 2, 12, 10]
d) []  e) None
Given the following Python code:

def q49(alist):
    '''(list) -> ??
    Test func.
    '''
    freqD = {}
    for item in alist:
        if item in freqD:
            freqD[item] += 1
        else:
            freqD[item] = 1
    return freqD

(49) Complete the type contract (replace ??):

a) int    b) str    c) list    d) dict    e) None

(50) When the following Python code is executed

```python
>>> q49(['a', 'b', 'c', 'c', 'a', 'd', 'e'])
```

the for loop in function q49 will execute ?? times:

a) 0    b) 1    c) 3    d) 5    e) 7

(51) What value is returned?

a) {'a': 2, 'b': 1, 'c': 2, 'd': 1, 'e': 1}

b) [[a', 2], [b', 1], [c', 2], [d', 1], [e', 1]]

c) {2: 'a', 1: 'b', 2: 'c', 1: 'd', 1: 'e'}

d) [[2, 'a'], [1, 'b'], [2, 'c'], [1, 'd'], [1, 'e']]

e) None

(52) Computational Problem Solving (Replace ?? with the correct response.)

a) looping    b) executing    c) printing    d) testing and debugging
Given the following Python code:

```python
def alphapinEncode(pin):
    ''' (int) -> str
    test func
    '''
    vowels = 'aeiou'
    consonants = 'bcdfghjklmnpqrstvwyz'
    tone = ''
    wkpin = pin
    while wkpin > 0:
        plain = wkpin % 100
        wkpin = wkpin // 100
        vowel = plain % 5
        vsound = vowels[vowel]
        cons = plain // 5
        csound = consonants[cons]
        tone = csound + vsound + tone
    return tone

>>> alphapinEncode(1234)
```

After the while loop is executed the first time (at line 23), the value of

(53) plain is ___34_____

(54) wkpin is ___12_____

(55) vsound is ___'u'_____

(56) tone is ___'ju'_____

(57) (4 pts.) What is the result of executing the following Python code:

```python
>>> for bit in '10':
    print(bit)  # True
    print(bit in '01')  # 0
    if bit not in '01':
        print(False)  # True
```
(58) (6 pts.) Write a function, isOdd, with one parameter, n, a positive integer. isOdd should return True if n is an odd number, and False otherwise. Code should be written according to CIS 210 style guidelines. Docstring examples of use should include at least one “normal” example and at least one “edge”, or “boundary”, example (for each example, comment on which type it is).

def isOdd(n):
    '''(int) -> bool

    Return True if n, a positive integer,
    is an odd number.
    
    >>> isOdd(5) #normal  
    True
    >>> isOdd(4) #normal
    False
    >>> isOdd(1) #edge
    True
    '''
    return n % 2 != 0

(59) (6 pts.) Write a function, addDigits, with one parameter, n, a positive integer. addDigits should return the sum of the digits in n, where \(100 \leq n \leq 999\). Use only numeric type operations (no strings, lists, etc.). Code should be written according to CIS 210 style guidelines. Docstring examples of use should include at least one “normal” example and at least one “edge”, or “boundary”, example (for each example, comment on which type it is).

def addDigits(n):
    '''(int) -> int

    Return sum of digits of n, a 3-digit number.
    
    >>> addDigits(123) #normal
    6
    >>> addDigits(100) #edge
    1
    >>> addDigits(999) #edge
    0
    '''
    digit_sum = 0
    for ctr in range(3):
        digit = n % 10
        n = n // 10
        digit_sum += digit
    return digit_sum