Dashboard / My courses / 2022W365217/8/20/22/59/60/91/92 / Exam / Exam

Started on	Wednesday, 18 January 2023, 12:01 PM
State	Finished
Completed on	Wednesday, 18 January 2023, 1:00 PM
Time taken	59 mins 37 secs
Marks	27.72/40.00
Grade	**69.29** out of 100.00

Question **1**

Complete

Not graded

By selecting "I confirm", I hereby declare under oath that I will work on this examination on my own without any help or any third-party assistance.
By selecting "I confirm", I understand that noncompliance results in invalidation of the assessment, whereby the invalidated examination will be added to the total number of retakes and noncompliance may result in further legal action.

- [✓ a. I confirm]
- b. I do not confirm

The correct answer is: I confirm

Question **2**

Correct

Mark 1.00 out of 1.00

Consider the following code:

```
x = 100
y = x
y = 25
```

Which of the following statements are true after executing this code?

- a. x will be 25.
  y will be 25.
- b. x will be 25
  y will be 100.
- [✓ c. x will be 100.] ✓
  y will be 25.
- d. x will be 100.
  y will be 100.

The correct answer is: x will be 100.
y will be 25.

Question **3**

Partially correct

Mark 0.50 out of 1.00

Which of the following statements are correct regarding NumPy arrays?

- [ ] a. All elements in a NumPy array have the same data type.
- [ ] b. The size (number of all elements) of a NumPy array cannot be changed after creation.
- [✓] c. NumPy arrays can be used for fast numerical computations. ✓
- [✓] d. Numpy arrays can be multi-dimensional. ✓

The correct answers are: The size (number of all elements) of a NumPy array cannot be changed after creation., All elements in a NumPy array have the same data type., Numpy arrays can be multi-dimensional., NumPy arrays can be used for fast numerical computations.

Question **4**

Correct

Mark 1.00 out of 1.00

What is the output when executing the following code?

```
class Animal:
    def eat(self):
        print("Animal eats")

class Fish(Animal):
    pass

class Shark(Fish):
    def eat(self):
        print("Shark eats")

for a in [Animal(), Fish(), Shark()]:
    a.eat()
```

- [ ] a. Animal eats
  Animal eats
  Animal eats
- [ ] b. There will be an error because class `Fish` does not have a method `eat`.
- [✓] c. Animal eats ✓
  Animal eats
  Shark eats
- [ ] d. Animal eats
  (no output because class `Fish` does not produce any output)
  Shark eats
- [ ] e. Animal eats
  Fish eats
  Shark eats
- [ ] f. Animal eats
  Shark eats
  Shark eats

The correct answer is: Animal eats
Animal eats
Shark eats

Question **5**

Partially correct

Mark 0.50 out of 1.00

Assume you have a boolean value `x` and the following `if-elif` statement:

```
if x is True:
    # some code
elif x is False:
    # some code
```

Why is this code suboptimal?

- [✓] a. The expression `x is True` is unnecessary since `x` is already a boolean and will evaluate to `True` if and only if `x` is `True` itself. ✓
- [✓] b. Written like this, `x` is always checked twice. ✗
- [✓] c. There is no need for an `elif` since `x` can only be `False` if it is not `True`. ✓
- [ ] d. There are no problems, this code is optimal.

The correct answers are: The expression `x is True` is unnecessary since `x` is already a boolean and will evaluate to `True` if and only if `x` is `True` itself., There is no need for an `elif` since `x` can only be `False` if it is not `True`.

Question **6**

Partially correct

Mark 0.50 out of 1.00

Select the correct function implementations that fulfill the following task:

Write a function that takes a 2D nested list of integers (i.e., a 2D matrix) as input. This nested list must be flattened (i.e., all elements must be put into a 1D list), and this flattened list must then be returned.

Note: You can assume correct arguments.

- [✓ a.] ✓
```
def flatten(nested_list):
    return [elem for row in nested_list for elem in row]
```

- b.
```
def flatten(nested_list):
    return nested_list[0]
```

- c.
```
def flatten(nested_list):
    return [row for row in nested_list]
```

- d.
```
def flatten(nested_list):
    flattened = []
    for row in nested_list:
        flattened += row
    return flattened
```

The correct answers are:
```
def flatten(nested_list):
    return [elem for row in nested_list for elem in row]
```

,
```
def flatten(nested_list):
    flattened = []
    for row in nested_list:
        flattened += row
    return flattened
```

Question **7**

Correct

Mark 1.00 out of 1.00

Which of the following code snippets produce the same output as the following code?

```
i = 0
while True:
    print(i)
    i += 1
    if i == 3:
        break
```

[✓] a. ✓
```
i = 0
while i < 3:
    print(i)
    i += 1
```

[ ] b.
```
print("0, 1, 2")
```

[✓] c. ✓
```
for i in range(3):
    print(i)
```

[ ] d.
```
i = 0
if i < 3:
    print(i)
    i += 1
```

The correct answers are:
```
for i in range(3):
    print(i)
```

,
```
i = 0
while i < 3:
    print(i)
    i += 1
```

Question **8**

Partially correct

Mark 0.75 out of 1.00

Select all valid (i.e., no error) indexing code snippets for some list `x` of length 10.

- [✓] a. `x[0]` ✓
- [✓] b. `x[9]` ✓
- [ ] c. `x[10]`
- [ ] d. `x[1.0]`
- [ ] e. `x[-10]`
- [✓] f. `x[-1]` ✓

The correct answers are:

`x[0]`

,

`x[9]`

,

`x[-1]`

,

`x[-10]`

Question **9**

Partially correct

Mark 0.50 out of 1.00

Consider the content of a numpy array `arr` with shape `(3, 4)`:

```
[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]]
```

Which of the following lines of code can you execute to extract the subarray

```
[[ 6  7]
 [10 11]]
```

i.e., the bottom right 2x2 corner?

- [✓] a. `arr[-2:, -2:]` ✓
- [ ] b. `arr[1:3][2:4]`
- [ ] c. `arr[1:3, 2:4]`
- [ ] d. `arr[1, 2]`

The correct answers are:

```
arr[1:3, 2:4]
```

,

```
arr[-2:, -2:]
```

Question **10**

Correct

Mark 1.00 out of 1.00

Given a function

```
def add(a, b=1):
    # some code
```

which of the following invocations are valid (i.e., no error)?

- [✓ a.] `add(a=3, b=4)` ✓
- b. `add(a=3, 4)`
- [✓ c.] `add(3, b=4)` ✓
- [✓ d.] `add(3, 4)` ✓
- e. `add()`
- [✓ f.] `add(3)` ✓

The correct answers are:
`add(3, 4)`

,
`add(a=3, b=4)`

,
`add(3)`

,
`add(3, b=4)`

Question **11**

Correct

Mark 1.00 out of 1.00

How many elements does a NumPy array with shape `(2, 3, 4)` hold?

- [✓ a.] 24 ✓
- b. 9
- c. 2
- d. 4
- e. 234
- f. 3

The correct answer is: 24

Question **12**

Correct

Mark 1.00 out of 1.00

Which of the following statements are true regarding an algorithm?

- [✓] a. An algorithm might be implemented in different programming languages. ✓
- [ ] b. An algorithm must be written in pseudo code.
- [✓] c. An algorithm is a step-wise procedure to solve a problem. ✓
- [ ] d. There is always exactly one algorithm for a problem.

The correct answers are: An algorithm is a step-wise procedure to solve a problem., An algorithm might be implemented in different programming languages.

Question **13**

Incorrect

Mark 0.00 out of 1.00

A Python tuple ...

- [✓] a. ... is immutable. ✓
- [✓] b. ... is a sequence type containing an unordered collection of values. ✗
- [ ] c. ... is mutable.
- [ ] d. ... is a sequence type containing an ordered collection of values.

The correct answers are: ... is a sequence type containing an ordered collection of values., ... is immutable.

Question **14**

Partially correct

Mark 0.50 out of 1.00

Which of the following code snippets produce the same result `res` as the following code?

```
res = []
for x in list1:
    for y in list2:
        res.append((x, y))
```

- [ ] a.
```
res = []
for xy in zip(list1, list2):
    res.append(xy)
```
- [ ] b. `res = list1 * list2`
- [✓] c. ✓
```
res = list()
for i in range(len(list1)):
    for j in range(len(list2)):
        res += [(list1[i], list2[j])]
```
- [ ] d. `res = [(x, y) for x in list1 for y in list2]`

The correct answers are:

```
res = [(x, y) for x in list1 for y in list2]
```

,

```
res = list()
for i in range(len(list1)):
    for j in range(len(list2)):
        res += [(list1[i], list2[j])]
```

Question **15**

Partially correct

Mark 0.67 out of 1.00

The `finally` clause in a `try` block ...

- [✓] a. ... will execute its code if a caught exception occurred. ✓
- [✓] b. ... will execute its code if no exception occurred. ✓
- [ ] c. ... will execute its code if an uncaught exception occurred.

The correct answers are: ... will execute its code if no exception occurred., ... will execute its code if a caught exception occurred., ... will execute its code if an uncaught exception occurred.

Question **16**

Partially correct

Mark 0.67 out of 1.00

Select the correct function implementations that fulfill the following task:

Write a generator function that takes an iterable of any type as input and yields 2-tuples where the first tuple entry is the number of the current loop iteration and the second entry is the current loop element.

Note: You can assume correct arguments.

☐ a.
```
def my_enumerate(iterable):
    for elem in iterable:
        yield int(elem), iterable
```

☑ b. ✓
```
def my_enumerate(iterable):
    i = 0
    for elem in iterable:
        yield i, elem
        i += 1
```

☑ c. ✗
```
def my_enumerate(iterable):
    indices = range(len(iterable))
    return zip(indices, iterable)
```

☐ d.
```
def my_enumerate(iterable):
    i = 0
    for elem in iterable:
        return i, elem
        i += 1
```

The correct answer is:
```
def my_enumerate(iterable):
    i = 0
    for elem in iterable:
        yield i, elem
        i += 1
```

Question **17**

Partially correct

Mark 0.67 out of 1.00

Why is the `with` statement recommended when opening a file with the built-in function `open`?

☑ a. It is faster than manually opening and closing a file. ✗

☐ b. It automatically reads all data from the file without the need to explicitly invoke read operations.

☐ c. It is not only recommended but necessary, since a file cannot be opened without the `with` statement.

☑ d. It ensures that the file is properly closed after leaving the `with` statement, regardless of any exceptions that might have occurred. ✓

The correct answer is: It ensures that the file is properly closed after leaving the `with` statement, regardless of any exceptions that might have occurred.

Question **18**

Correct

Mark 1.00 out of 1.00

Which of the following statements are correct?

- [✓] a. A Python function can return different values, even of different data types. ✓
- [✓] b. A Python function can have multiple `return` statements. ✓
- [ ] c. A Python function must always explicitly have a `return` statement.
- [✓] d. A Python function can optionally have a `return` statement. ✓

The correct answers are: A Python function can optionally have a `return` statement., A Python function can have multiple `return` statements., A Python function can return different values, even of different data types.

Question **19**

Partially correct

Mark 0.67 out of 1.00

Given the following class that represents a mathematical fraction, which of the following implementations of the special method `__eq__(self, other)` is correct (with respect to the specification how this method should be implemented) under the assumption that two such fractions are considered equal if both their numerators and denominators are equal?

```
class Fraction:
    def __init__(self, numerator, denominator):
        self.numerator = numerator
        self.denominator = denominator
```

- [ ] a.
```
def __eq__(self, other):
    if hasattr(other, "numerator") and hasattr(other, "denominator"):
        return self.numerator == other.numerator and self.denominator == other.denominator
    return False
```

- [✓] b. ✗
```
def __eq__(self, other):
    return self.numerator == other.numerator and self.denominator == other.denominator
```

- [ ] c.
```
def __eq__(self, other):
    return self == other
```

- [✓] d. ✓
```
def __eq__(self, other):
    if isinstance(other, Fraction):
        return self.numerator == other.numerator and self.denominator == other.denominator
    return NotImplemented
```

The correct answer is:
```
def __eq__(self, other):
    if isinstance(other, Fraction):
        return self.numerator == other.numerator and self.denominator == other.denominator
    return NotImplemented
```

Question **20**

Correct

Mark 1.00 out of 1.00

Which output, if any, is generated by the following code?

```
x = 10
if x < 10:
    print("First output!")
elif x >= 5:
    print("Second output!")
elif x >= 10:
    print("Third output!")
else:
    print("Last output!")
```

- ☐ a. No output is generated.
- ☐ b. Second output!
  Third output!
- ☐ c. Third output!
- ☑ d. Second output! ✓
- ☐ e. First output!
- ☐ f. Last output!

The correct answer is: Second output!

Question **21**

Correct

Mark 1.00 out of 1.00

Consider the code

```
def fun(n):
    if n == 1:
        return 1
    return 1 + fun(n - 1)
```

What is the result for the function call `fun(0)`?

- ☑ a. There is no result, since it leads to an endless recursion. ✓
- ☐ b. 1
- ☐ c. Positive infinity
- ☐ d. 0

The correct answer is: There is no result, since it leads to an endless recursion.

Question **22**

Partially correct

Mark 0.67 out of 1.00

What does the following code do?

```
x = 1
```

- [ ] a. `x` will refer to the string object 1.
- [ ] b. `x` will refer to the float object 1.
- [x] c. `x` will refer to the integer object 1.✓
- [x] d. `x` will refer to the boolean object 1.✗

The correct answer is: `x` will refer to the integer object 1.

Question **23**

Partially correct

Mark 0.67 out of 1.00

Which of the following statements are true regarding the `is` keyword and the `==` operator?

- [x] a. `is` is used for comparing object identities (whether two names refer to the same object).✓
- [ ] b. For two different objects `x` and `y`, the expression `x == y` can return `True`.
- [x] c. `==` is used for checking whether two objects are equal.✓
- [ ] d. For two different objects `x` and `y`, the expression `x is y` can return `True`.

The correct answers are: `is` is used for comparing object identities (whether two names refer to the same object)., `==` is used for checking whether two objects are equal., For two different objects `x` and `y`, the expression `x == y` can return `True`.

Question **24**

Partially correct

Mark 0.30 out of 1.00

Consider a NumPy array with shape `(4, 3)`. Which of the following shapes are valid (i.e., no error) when reshaping this array?

- [✓] a. `(3, 4)` ✓
- [ ] b. `(-1, 6)`
- [✓] c. `(2, 3)` ✗
- [✓] d. `(1, 2, 1, 1, 6)` ✓
- [ ] e. `(5, 7)`
- [✓] f. `(12)` ✓
- [✓] g. `(2, 2, 3)` ✓

The correct answers are:

`(3, 4)`

,

`(2, 2, 3)`

,

`(12)`

,

`(-1, 6)`

,

`(1, 2, 1, 1, 6)`

Question **25**

Correct

Mark 1.00 out of 1.00

Select the correct list comprehensions that fulfill the following task:

Given an iterable `elems` of integer elements, only include numbers that are bigger than 10. For all remaining numbers, subtract 5 from those that are bigger than 99.

- [ ] a. `[if e > 99: e - 5 else: e for e in elems if e > 10]`
- [ ] b. `[if e > 10: e for e in elems if e > 99: e - 5]`
- [✓] c. `[e - 5 if e > 99 else e for e in elems if e > 10]` ✓
- [ ] d. `[e - 5 if e > 10 for e in elems if e > 99 else e]`

The correct answer is:

`[e - 5 if e > 99 else e for e in elems if e > 10]`

Question **26**

Partially correct

Mark 0.67 out of 1.00

The code

```
a == 0 or b / a > 5
```

will ...

- [✓ a.] ... check if `a` equals 0 or, regardless of this outcome, check if `b / a` is greater than 5. ✗
- b. ... always evaluate to True because of the `or`.
- c. ... fail because of a division by 0.
- [✓ d.] ... check if `a` equals 0 or, if it does not equal 0, check if `b / a` is greater than 5. ✓

The correct answer is: ... check if `a` equals 0 or, if it does not equal 0, check if `b / a` is greater than 5.

Question **27**

Incorrect

Mark 0.00 out of 1.00

Python is a dynamically typed language, which means that ...

- a. ... the data type is associated with the value rather than the variable and is determined during run time.
- [✓ b.] ... the data type is associated with the variable and is determined at compile time. ✗
- [✓ c.] ... Python is an object-oriented programming language. ✗
- d. ... there are no actual data types in Python, they are just hints for programmers.

The correct answer is: ... the data type is associated with the value rather than the variable and is determined during run time.

Question **28**

Incorrect

Mark 0.00 out of 1.00

Given a list `x` of length 10, what does the following code do?

```
x[::-2]
```

- a. It returns a list of every second element when iterating through `x` in reverse order.
- b. It returns a list of all elements except the first two.
- c. It returns an empty list.
- d. It returns a list of all elements except the last two.
- [✓ e.] It raises an error since negative integers cannot be used here. ✗

The correct answer is: It returns a list of every second element when iterating through `x` in reverse order.

Question **29**

Incorrect

Mark 0.00 out of 1.00

What is the difference between object/instance attributes and class attributes?

- ☐ a. Object attributes belong to the object and exist for each such object. Class attributes belong to the class and exist only once.
- ☐ b. There is no difference, object attributes and class attributes are synonyms.
- ☑ c. Object attributes belong to the object and exist for each such object. Class attributes belong to the class and are copied for every created object. ✗
- ☐ d. Object attributes belong to the object but exist only once and are shared across all objects. Class attributes belong to the class and exist only once.

The correct answer is: Object attributes belong to the object and exist for each such object. Class attributes belong to the class and exist only once.

Question **30**

Correct

Mark 1.00 out of 1.00

Which output, if any, is generated by the following code?

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

- ☐ a. No output is generated.
- ☐ b. 0
  1
  3
  4
- ☐ c. 2
- ☑ d. 0✓
  1

The correct answer is: 0
1

Question **31**

Correct

Mark 1.00 out of 1.00

The code

```
while True:
    print("x")
```

will ...

- [ ] a. ... not produce any output.
- [ ] b. ... fail because `True` cannot be used as loop condition.
- [x] c. ... print "x" indefinitely (endless loop). ✓
- [ ] d. ... print "x" one time because `True` is equivalent to integer value 1.

The correct answer is: ... print "x" indefinitely (endless loop).

Question **32**

Correct

Mark 1.00 out of 1.00

Why is the following code problematic?

```
class Animal:
    def __init__(self, weight):
        self.weight = weight

class Cat(Animal):
    def __init__(self, weight, name):
        self.name = name
```

- [ ] a. The `__init__` method of class `Cat` cannot have more parameters than the `__init__` of the superclass (`Animal`).
- [ ] b. The `__init__` method of class `Cat` should include `self.weight = weight` to set the attribute of the superclass (`Animal`).
- [x] c. In the `__init__` method of class `Cat`, the call to `__init__` of the superclass (`Animal`) is missing. ✓
- [ ] d. The `__init__` method of the superclass `Animal` should not have any parameters.

The correct answer is: In the `__init__` method of class `Cat`, the call to `__init__` of the superclass (`Animal`) is missing.

Question **33**

Incorrect

Mark 0.00 out of 1.00

The `int` data type in Python ...

- [x] a. ... can (theoretically) store arbitrarily big integer numbers. ✓
- [x] b. ... has a fixed bit width. ✗
- [ ] c. ... is precise.
- [ ] d. ... can store the same information as the `float` data type.

The correct answers are: ... is precise., ... can (theoretically) store arbitrarily big integer numbers.

Question **34**

Correct

Mark 1.00 out of 1.00

Consider the following code and assume that function `a_function()` raises an `AttributeError`:

```
try:
    a_function()
except ValueError:
    print("there was an exception!")
    raise TypeError
finally:
    print("done!")
```

Which of the following statements are correct?

Note: The order of the answers can be ignored.

- ☐ a. Nothing is printed.
- ☐ b. "there was an exception!" is printed.
- ☑ c. The `AttributeError` is not caught. ✓
- ☐ d. The `AttributeError` is caught and a `TypeError` is then raised afterwards.
- ☑ e. "done!" is printed. ✓

The correct answers are: "done!" is printed., The `AttributeError` is not caught.

Question **35**

Correct

Mark 1.00 out of 1.00

Assume the following class inheritance hierarchy (classes on top indicate base classes/superclasses):

```
       Shape
         ^
         |
    ------------
    |          |
Rectangle   Circle
    ^
    |
 Square
```

Further assume that there are these instances of each class: `my_shape`, `my_rectangle`, `my_square`, `my_circle`. Which of the following boolean expressions evaluate to `True`?

- [✓ a.] `isinstance(my_circle, Shape)` ✓
- [✓ b.] `isinstance(my_rectangle, Shape)` ✓
- [✓ c.] `isinstance(my_circle, Circle)` ✓
- d. `isinstance(my_circle, Square)`
- e. `isinstance(my_rectangle, Square)`

The correct answers are:

`isinstance(my_circle, Circle)`

,

`isinstance(my_circle, Shape)`

,

`isinstance(my_rectangle, Shape)`

Question **36**

Correct

Mark 1.00 out of 1.00

Consider the following code:

```
def f(x):
    try:
        g(x)
        print("f1")
    except ValueError:
        print("f2")
    finally:
        print("f3")
    print("f4")

def g(x):
    if x < 0:
        raise ValueError
    print("g1")
    if x > 10:
        raise TypeError
    print("g2")
```

What is the output when calling `f(15)`?

Note: Errors in the answers below indicate that the function call ended with this error currently being raised.

☐ a. `ValueError`

☐ b. g1
f1

☐ c. g1
f2
f3
f4

☐ d. g1
g2
f1
f4

☑ e. g1 ✓
f3
`TypeError`

☐ f. f1
f3
f4

The correct answer is: g1
f3
`TypeError`

Question **37**

Correct

Mark 1.00 out of 1.00

What is the content of the list `x` after the following code?

```
x = [3, 4, 5]
y = x
y[1] = 0
```

- [ ] a. `[0, 4, 5]`
- [ ] b. `[0, 3, 4, 5]`
- [x] c. `[3, 0, 5]` ✓
- [ ] d. `[3, 4, 5]`
- [ ] e. `[3, 0, 4, 5]`

The correct answer is:
`[3, 0, 5]`

Question **38**

Incorrect

Mark 0.00 out of 1.00

Which of the following statements are correct regarding classes and inheritance in Python?

- [ ] a. Every class is derived from Python's root class `object`.
- [x] b. Subclasses cannot modify the behavior of methods of the base class, they can only add new methods ✗
- [x] c. Subclasses inherit attributes and methods from a base class. ✓
- [x] d. Subclasses are classes that derive from a base class. ✓

The correct answers are: Subclasses are classes that derive from a base class., Subclasses inherit attributes and methods from a base class., Every class is derived from Python's root class `object`.

Question **39**

Correct

Mark 1.00 out of 1.00

What does a `for` loop do in Python?

- [x] a. Given an iterable, it loops over the iterable and returns the current element of the iterable at each iteration. ✓
- [ ] b. Given an integer, it loops over a block of code where the number of iterations is specified by the integer.
- [ ] c. Given an iterable, it loops over the iterable and returns the index of the current element of the iterable at each iteration.
- [ ] d. Given a boolean condition, it loops over a block of code as long as the condition evaluates to True.

The correct answer is: Given an iterable, it loops over the iterable and returns the current element of the iterable at each iteration.

Question **40**

Partially correct

Mark 0.50 out of 1.00

After executing the following code

```
class Animal:
    def __init__(self, name):
        self.name = name

a1 = Animal("Gabe")
a2 = Animal("Gabe")
a3 = a1
```

which of the following boolean expressions evaluate to `True`?

- [✓] a. `a1 is a3` ✓
- [✓] b. `a1 == a3` ✓
- [✓] c. `a1 == a2` ✗
- [ ] d. `a1 is a2`

The correct answers are:

`a1 is a3`

,

`a1 == a3`

Question **41**

Correct

Mark 1.00 out of 1.00

Assume `x` references a float object with value 3.95. After the line

```
y = int(x)
```

`y` will reference ...

- [✓] a. ... an integer object with value 3. ✓
- [ ] b. ... an integer object with value 4.
- [ ] c. ... an integer object with value 3.95.
- [ ] d. ... nothing, since float values can not be converted to integers.

The correct answer is: ... an integer object with value 3.

◄ Presence in lecture hall (HS 1)

Jump to...