Jeremias Lacanienta

PCAP
Certified Associate in Python
Programming Certification

PCAP-31-03
Practice Exam

Exam block #1: Modules and Packages (12%)

Objectives covered by the block (6 items)

* import variants; advanced qualifying for nested modules

What is the output of the following code if **spam.py** is run?

 # spam.py

```
print("spam", end=' ')
import ham
```

 # ham.py

```
import eggs
print("ham", end=' ')
```

 # eggs.py

```
print("eggs", end=' ')
```

( ) Syntax Error
( ) spam eggs ham
( ) spam ham
( ) eggs ham spam
( ) spam ham eggs

Answer >>>

How do you call the function ham() saved as **spam.py** below?

```
def ham():
    print("Hello World")
```

[ ] import spam; ham()
[ ] import spam.ham; ham()
[ ] import spam; spam.ham()
[ ] from spam import ham; ham()
[ ] import ham from spam; ham()

Answer >>>

* import variants; advanced qualifying for nested modules

Given the following package layout

```
package/
   subpackage1/
      __init__.py
      moduleX.py
      moduleY.py
   subpackage2/
      moduleZ.py
   moduleA.py
```

Select all option(s) containing valid relative imports called from __init__.py

[ ] from .moduleY import spam
[ ] from .moduleY import spam as ham
[ ] from ..subpackage1 import moduleY
[ ] from ..subpackage2.moduleZ import eggs
[ ] from ..moduleA import foo

Answer >>>

How will you shorten the function call to spam() defined inside packageA.subpackageB.subpackageC.moduleD?

[ ] import packageA.subpackageB.subpackageC.moduleD
[ ] import packageA.subpackageB.subpackageC.moduleD as p
[ ] import packageA.subpackageB.subpackageC.moduleD alias p
[ ] from packageA.subpackageB.subpackageC.moduleD import *
[ ] from packageA.subpackageB.subpackageC.moduleD import spam
[ ] from packageA.subpackageB.subpackageC.moduleD import spam as s
[ ] from packageA.subpackageB.subpackageC.moduleD import spam alias s

Answer >>>

* dir(); sys.path variable

Select all valid parameters to function dir()

[ ] No parameter
[ ] Object
[ ] 0
[ ] None

Answer >>>

Select all valid option(s) about the result of dir()
[ ] A list of filenames inside the directory
[ ] A list of the module's attribute
[ ] A list of names of class attributes
[ ] A list of names of object attributes
[ ] A list of names of the base class attributes
Answer >>>

***** dir(); sys.path variable

Select all valid option(s) about sys.path
[ ] sys.path is a string that specifies the path where Python is installed
[ ] sys.path is a string that specifies the path of the compiled Python bytecode
[ ] sys.path is a list of strings that specifies the search path for modules
[ ] A program is free to modify sys.path for its own purpose.
Answer >>>

***** math: ceil(), floor(), trunc(), factorial(), hypot(), sqrt(); random: random(), seed(), choice(), sample()

What is the output of the following code?

```
>>> math.ceil(-1.1)
```

( ) -1
( ) -1.0
( ) -2
( ) -2.0
Answer >>>

***** math: ceil(), floor(), trunc(), factorial(), hypot(), sqrt(); random: random(), seed(), choice(), sample()

What is the output of the following code?

```
>>> math.floor(-1.1)
```

( ) -1
( ) -1.0

( ) -2
( ) -2.0

Answer >>>

* math: ceil(), floor(), trunc(), factorial(), hypot(), sqrt(); random: random(), seed(), choice(), sample()

What is the output of the following code?

```
>>> math.factorial(3.0))
```

( ) 6
( ) 6.0
( ) TypeError: type float doesn't define __factorial__ method
( ) TypeError: factorial() takes 2 arguments

Answer >>>

What is the output of the following code?

```
>>> math.factorial(-3.0)
```

( ) -6
( ) -6.0
( ) TypeError: type float doesn't define __factorial__ method
( ) ValueError: factorial() not defined for negative values

Answer >>>

* math: ceil(), floor(), trunc(), factorial(), hypot(), sqrt(); random: random(), seed(), choice(), sample()

What is the output of the following code?

```
>>> math.hypot(2)
```

( ) 3.6055512754639896
( ) 2.0
( ) TypeError: type int doesn't define __hypot__ method
( ) TypeError: hypot() takes 2 arguments

Answer >>>

* math: ceil(), floor(), trunc(), factorial(), hypot(), sqrt(); random: random(), seed(), choice(), sample()

What is the output of the following code?

```
>>> math.sqrt(1)
```

( ) 0.5
( ) 1
( ) 1.0
( ) TypeError: type int doesn't define __sqrt__ method

Answer >>>

***** math: ceil(), floor(), trunc(), factorial(), hypot(), sqrt(); random: random(), seed(), choice(), sample()

Select all option(s) which returns a random floating number between 0 and 1?

( ) math.random()
( ) math.random(1.0)
( ) random.random()
( ) random.random(1.0)

Answer >>>

Select all option(s) which returns a random number between 0 and 100?

( ) random.random(100)
( ) random.random(0, 100)
( ) random.random()*100
( ) random.random(100.0)

Answer >>>

***** math: ceil(), floor(), trunc(), factorial(), hypot(), sqrt(); random: random(), seed(), choice(), sample()

What can be the possible output of the following code?

```
random.seed(10, 2)
print(random.random())
```

( ) 3.6055512754639896
( ) 0.5714025946899135
( ) AttributeError: module 'random' has no attribute 'seed'
( ) TypeError: seed() takes 1 argument

Answer >>>

* math: ceil(), floor(), trunc(), factorial(), hypot(), sqrt(); random: random(), seed(), choice(), sample()

Select all option(s) to properly call the choice() and/or choices() function?

[ ] random.choice("spam", "ham", "eggs")
[ ] random.choice(["spam", "ham", "eggs"])
[ ] random.choice({"spam", "ham", "eggs"})
[ ] random.choices(["spam", "ham", "eggs"])
[ ] random.choices(["spam", "ham", "eggs"], weights = [10, 1, 1], k = 14)

Answer >>>

* math: ceil(), floor(), trunc(), factorial(), hypot(), sqrt(); random: random(), seed(), choice(), sample()

What can be the possible output of the following code?

```
>>> random.sample(["spam", "ham", "eggs"], k = 1)
```

( ) spam
( ) [spam]
( ) TypeError: sample() got an unexpected keyword argument 'k'
( ) TypeError: sample() takes 1 argument

Answer >>>

* platform: platform(), machine(), processor(), system(), version(), python_implementation(), python_version_tuple()

Select all option(s) to properly call the platform() function?

[ ] system.platform()
[ ] platform.platform()
[ ] system.platform(aliased=0, terse=0)
[ ] platform.platform(alias=0, version=0)
[ ] platform.platform(aliased=0, terse=0)

Answer >>>

* platform: platform(), machine(), processor(), system(), version(), python_implementation(), python_version_tuple()

Select all option(s) to properly call the machine() function?

[ ] system. machine()

[ ] platform. machine()
[ ] system. machine(aliased=0)
[ ] platform. machine (terse=0)
[ ] platform. machine (None)
Answer >>>

***** platform: platform(), machine(), processor(), system(), version(), python_implementation(), python_version_tuple()

Select all option(s) to properly call the processor() function?
[ ] system.processor()
[ ] platform.processor()
[ ] system.processor(aliased=0)
[ ] platform.processor(terse=0)
[ ] platform.platform(None)
Answer >>>

***** platform: platform(), machine(), processor(), system(), version(), python_implementation(), python_version_tuple()

Select all option(s) to properly call the system() function?
[ ] system.system()
[ ] platform.system()
[ ] system.system(aliased=0)
[ ] platform.system(terse=0)
[ ] platform.system(None)
Answer >>>

Select all valid option(s) about system() function
[ ] system() returns the OS hosting Python
[ ] system() returns the execution environment of Python
[ ] Possible return values are **Linux**, **Darwin**, **Java**, **Windows** or an empty string if it can't be determined.
[ ] Possible return values are **CPython**, **IronPython**, **Jython**, **PyPy**.
Answer >>>

***** platform: platform(), machine(), processor(), system(), version(), python_implementation(), python_version_tuple()

Select all option(s) to properly call the version() function?

[ ] system.version()
[ ] platform.version()
[ ] system.version(aliased=0)
[ ] platform.version(terse=0)
[ ] platform.version(None)

Answer >>>

What is the datatype of the return value of the function platform.version()?

( ) int
( ) float
( ) str
( ) array

Answer >>>

***** platform: platform(), machine(), processor(), system(), version(), python_implementation(), python_version_tuple()

Select all option(s) to properly call the python_implementation() function?

[ ] system.python_implementation()
[ ] platform.python_implementation()
[ ] system.python_implementation(aliased=0)
[ ] platform.python_implementation(terse=0)
[ ] platform.python_implementation(None)

Answer >>>

Select all option(s) about the python_implementation() that is TRUE?

[ ] python_implementation() returns the OS hosting Python
[ ] python_implementation() returns the execution environment of Python
[ ] Possible return values are **Linux**, **Darwin**, **Java**, **Windows** or an empty string if it can't be determined.
[ ] Possible return values are **CPython**, **IronPython**, **Jython**, **PyPy**.

Answer >>>

***** platform: platform(), machine(), processor(), system(), version(), python_implementation(), python_version_tuple()

Select all option(s) to properly call the python_version_tuple() function?

[ ] system.python_version_tuple()
[ ] platform.python_version_tuple()
[ ] system.python_version_tuple(aliased=0)
[ ] platform.python_version_tuple(terse=0)
[ ] platform.python_version_tuple(None)

Answer >>>

***** idea: __pycache__, __name__, public variables, __init__.py

Which of the statements below is valid?

[ ] Python is interpreted therefore it never compiles the **py** files.
[ ] Python is interpreted however it compiles the **py** file into **pyc** file.
[ ] Compiled Python files is stored inside the **__pyc__** folder
[ ] Compiled Python files is stored inside the **__pycache__** folder
[ ] Compiled Python files is stored inside the **__cache__** folder

Answer >>>

The extension of a compiled bytecode of the Python source file is

( ) .py
( ) .pyc
( ) .__pycache__
( ) Python is an interpreted language hence it does not compile the source file

Answer >>>

***** idea: __pycache__, __name__, public variables, __init__.py

Select all valid option(s) about __name__

[ ] The __name__ is a built-in constant and can't be modified
[ ] The __name__ is a built-in variable and can be modified
[ ] The __name__ by default is None and must be set
[ ] If the source is the main program, the interpreter sets __name__ to "__main__"
[ ] If the file is imported from another module, __name__ will be set with the module's name.

Answer >>>

***** idea: __pycache__, __name__, public variables, __init__.py

How should you write the variable spam to inform a module user that it should not be accessed directly?

[ ] spam since all variables in modules are considered private
[ ] _spam
[ ] __spam
[ ] SPAM

Answer >>>

***** idea: __pycache__, __name__, public variables, __init__.py

Select all valid option(s) about __init__.py

[ ] __init__.py is contained in regular packages
[ ] __init__.py is contained in namespace packages
[ ] __init__.py is automatically executed when the regular package is imported.
[ ] __init__.py is automatically executed when the namespace package is imported

Answer >>>

***** searching for modules/packages; nested packages vs directory tree

What directories are searched by the interpreter for spam.py given the code below?

```
import spam
print(spam.ham)
print(spam.eggs)
```

[ ] Directory where spam.py was run
[ ] Current directory if the interpreter is run interactively
[ ] List of directories contained in PATH environment variable
[ ] List of directories contained in PYTHONPATH environment variable
[ ] Python installation-dependent directories configured during installation
[ ] List of directories in sys.path

Answer >>>

Exam block #2: Exceptions (14%)

Objectives covered by the block (5 items)

* except, except:-except; except:-else:, except (e1,e2)

What is the output of the following code?

```
try:
    abcd
    efgh
except:
    pass
```

( ) No output
( ) SyntaxError: invalid syntax
( ) NameError: name 'UndefinedException' is not defined
( ) Add () on Line 2 and 3 to fix the syntax error

Answer >>>

What is the output of the following code?

```
>>> try:
...     raise OSError
... finally:
...     pass
```

( ) No output
( ) OSError
( ) NameError: name 'OSError' is not defined
( ) SyntaxError: invalid syntax

Answer >>>

What is the output of the following code?

```
try:
    raise ValueError
except TypeError, ValueError:
    raise
```

( ) No output
( ) TypeError
( ) ValueError
( ) SyntaxError: invalid syntax

Answer >>>

* except, except:-except; except:-else:, except (e1,e2)

What will happen if spam.py is run?

 # spam.py

```
try:
    print(x)
except:
    print("An exception occurred")
```

( ) the script will run but will not print anything
( ) **None** will be printed
( ) **An exception occurred** will printed
( ) Compile time error

Answer >>>

What is the output of the following code?

```
>>> def this_fails():
...     x = 1/0
>>> try:
...     this_fails()
... except ZeroDivisionError:
...     pass
```

( ) No output
( ) SyntaxError: invalid syntax
( ) ZeroDivisionError: division by zero
( ) NameError: name 'ZeroDivisionError' is not defined

Answer >>>

* except, except:-except; except:-else:, except (e1,e2)

Which option(s) will print **ELSE** given the following code?

```
try:
    <<< INSERT CODE HERE >>>
except ZeroDivisionError:
    print('ZeroDivisionError')
except TypeError:
    print('TypeError')
else:
    print('ELSE')
```

[ ] raise Exception
[ ] raise ZeroDivisionError
[ ] raise TypeError
[ ] raise
[ ] pass
[ ] replace <<< INSERT CODE HERE >>> with blank

Answer >>>

What is the output of the following code?

```
try:
    print("1", end='')
    raise Exception
    print("2", end='')
except BaseException:
    print("3", end='')
else:
    print("4", end='')
finally:
    print("5")
```

( ) NameError: name 'BaseException' is not defined
( ) 1235
( ) 1245
( ) 135
( ) 145

Answer >>>

What is the output of the following code?

```
class E(Exception):
    def __init__(self, message):
        self.message = message
    def __str__(self):
        return "Surprise"

try:
    raise Exception("Stop")
except E as e:
    print(e)
else:
    print("Goodbye")
```

( ) Unhandled Exception
( ) Surprise
( ) Stop
( ) Goodbye

Answer >>>

***** except, except:-except; except:-else:, except (e1,e2)

What is the output of the following code?

```
try:
    raise Exception
except:
    print("Spam", end='')
except BaseException:
    print("Ham", end='')
except Exception:
    print("Eggs")
```

( ) Eggs
( ) Spam Eggs
( ) Spam Ham Eggs

( ) Syntax Error

Answer >>>

If there are more than 1 except clause, what happens after a try clause executes?

[ ] None of the except is executed
[ ] At least 1 except is executed
[ ] Not more than 1 except is executed
[ ] Exactly 1 of the except is executed

Answer >>>

* the hierarchy of exceptions

Which of the statements below is valid?

[ ] The finally branch in a try block is always executed.
[ ] The finally branch in a try block will only be executed if an exception occurs.
[ ] The finally branch in a try block will only be executed if the exception did not occur
[ ] The finally branch in a try block is optional
[ ] The finally branch in a try block is required because it is always executed.

Answer >>>

What is the output of the following code?

```
class Spam(Exception):
    pass
class Ham(Spam):
    pass
for cls in [Spam, Ham]:
    try:
        raise cls()
    except Spam:
        print("Spam", end=" ")
    except Ham:
        print("Ham", end=" ")
```

( ) Spam Ham

( ) Spam Spam
( ) Spam Ham Spam Ham
( ) Invalid Syntax

Answer >>>

Which option are valid replacements for the marker in the given code?

```
>>> try:
...     x = 1/0
... <<< INSERT CODE HERE >>>
...     pass
```

[ ] except BaseException:
[ ] except Exception:
[ ] except MathError:
[ ] except ArithmeticException:
[ ] except ArithmeticError:
[ ] except DivisionZeroError:
[ ] except ZeroDivisionError:

Answer >>>

* raise, raise ex, assert

What is the output of the following code if spam.txt does not exist?

```
import sys
try:
    f = open('spam.txt')
    s = f.readline()
except:
    raise
```

( ) the script will run but will not print anything
( ) "None" will be printed
( ) FileNotFoundError: [Errno 2] No such file or directory: 'spam.txt'
( ) Compile time error

Answer >>>

* raise, raise ex, assert

Select which option will call the __init__ method of Exception based on the

code below.

```
class SpamException(Exception):
    def __init__(self, message):
        <<< INSERT CODE HERE >>>
        self.message = message
raise SpamException("Spam")
```

[ ] super().__init__(message)
[ ] Exception.__init__(self, message)
[ ] super(SpamException, self).__init__(message)
[ ] super.__init__(message)

Answer >>>

What is the output of the following code?

```
try:
    raise UndefinedException
except:
    pass
```

[ ] No output
[ ] SyntaxError: invalid syntax
[ ] NameError: name 'UndefinedException' is not defined
[ ] Add () on Line 2 to fix the syntax error

Answer >>>

What is the output of the following code?

```
try:
    raise UndefinedException
except NameError:
    print('NameError')
except UndefinedException:
    print('UndefinedException')
except:
    pass
```

( ) No output
( ) NameError

( ) UndefineException
( ) SyntaxError: invalid syntax

Answer >>>

What is the output of the following code?

```
try:
    raise IOError
except IOError:
    raise RuntimeError from None
```

( ) No output
( ) IOError
( ) RuntimeError
( ) SyntaxError: invalid syntax

Answer >>>

What is the output of the following code?

```
try:
    raise IOError
except IOError as e:
    raise RuntimeError from e
```

( ) No output
( ) IOError
( ) RuntimeError
( ) SyntaxError: invalid syntax

Answer >>>

***** raise, raise ex, assert

Which of the statements below is valid?

```
spam = 0
assert spam == 0
```

( ) AssertionError will be triggered because the expression is True
( ) No AssertionError will be triggered since the expression is True
( ) Missing parentheses in call to assert error will be displayed
( ) The word True will be printed on screen

Answer >>>

What is the result of the following code?

```
>>> assert(False, 'Trigger Assertion')
```

[ ] No output
[ ] Trigger Assertion
[ ] SyntaxError: invalid syntax
[ ] Assertion is always true

Answer >>>

***** event classes, except E as e, arg property
***** event classes, except E as e, arg property

Which option(s) are valid except clause for ZeroDivisionError to be accessed as variable e?

[ ] except ZeroDivisionError as e:
[ ] except ZeroDivisionError(e):
[ ] except (ZeroDivisionError) as e:
[ ] except ZeroDivisionError e:
[ ] except (ZeroDivisionError as e):

Answer >>>

What is the output of the following code?

```
try:
    a = 1/'0'
except (ZeroDivisionError, TypeError) as e:
    print(type(e))
```

( ) The script will run but will not print anything
( ) <class 'ZeroDivisionError'>
( ) <class 'TypeError'>
( ) Invalid Syntax

Answer >>>

***** event classes, except E as e, arg property

Which option will print ('spam', 'eggs') based on the following code?

```
try:
    raise Exception('spam', 'eggs')
except Exception as exception:
```

```
    <<< INSERT CODE HERE >>>
```

[ ] print(exception.params)
[ ] print(exception)
[ ] print(exception.args)
[ ] print(exception.iterable[:])

Answer >>>

What is the output of the following code?

```
>>> type(Exception().args)
```

( ) <class 'str'>
( ) <class 'list'>
( ) <class 'tuple'>
( ) <class 'dict'>

Answer >>>

What is the output of the following code?

```
>>> try:
...     raise Exception('spam', 'eggs')
... except Exception as inst:
...     x, y = inst.args
>>> x, y
```

( ) ('spam', 'eggs')
( ) ValueError: too many values to unpack (expected 2)
( ) TypeError: 'tuple' object does not support item assignment
( ) SyntaxError: invalid syntax

Answer >>>

***** self-defined exceptions, defining and using

What is the output of the following code?

```
class AgeException(Exception):
    def __init__(self, age):
        super(AgeException, self).__init__("AgeException")
try:
    raise AgeException(16)
```

```
except AgeException as e:
    print(e)
```

( ) The script will run but will not print anything
( ) **AgeException** will be printed
( ) TypeError: super() argument 1 must be type
( ) TypeError:__init__() argument 1 must be type

Answer >>>

***** self-defined exceptions, defining and using

What is the output of the following code?

```
class MyException(Exception):
    pass
try:
    raise MyException("spam", "ham", "eggs")
except MyException as s:
    print(s)
```

( ) The script will run but will not print anything
( ) spam ham eggs
( ) ('spam', 'ham', 'eggs')
( ) TypeError: expected Exception not type

Answer >>>

Exam block #3: Strings (18%)

Objectives covered by the block (8 items)

* ASCII, UNICODE, UTF-8, codepoints, escape sequences

Select all valid option(s) below about string
[ ] string.ascii_letters is a concatenation of ascii_lowercase and ascii_uppercase
[ ] string.ascii_letters is a concatenation of ascii_lowercase, ascii_uppercase and digits
[ ] string.ascii_letters are all printable characters found in the keyboard
[ ] string.ascii_lowercase contains 'abcdefghijklmnopqrstuvwxyz'
[ ] string.ascii_uppercase contains 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
Answer >>>

* ASCII, UNICODE, UTF-8, codepoints, escape sequences
* ASCII, UNICODE, UTF-8, codepoints, escape sequences
* ASCII, UNICODE, UTF-8, codepoints, escape sequences
* ASCII, UNICODE, UTF-8, codepoints, escape sequences

What is the output of the following code?

```
print("\\\\")
```

( ) Syntax Error
( ) \\\\
( ) \\\
( ) \\
( ) \
Answer >>>

What is the output of the following code?

```
print("\\\\\")
```

( ) Syntax Error
( ) \\\\\
( ) \\\\

( ) \\\
( ) \\
( ) \

Answer >>>

What is the output of the following code?

```
print("C:\Program Files\Microsoft\Windows NT", end="")
print("\")
```

( ) Syntax Error
( ) C:\Program Files\Microsoft\Windows NT\
( ) Replace escaped characters with "?" e.g. C:?rogram Files?icrosoft?indows NT?
( ) Ignore escaped characters e.g. C: rogram Filesicrosoftindows NT

Answer >>>

What is the output of the following code?

```
print("\\/\\//", len("\\/\\//"))
```

( ) \\/\\// 8
( ) \/\// 8
( ) \/\// 6
( ) //// 4
( ) Syntax Error

Answer >>>

What will be printed in the following code?

```
spam = """"""""
ham = """
"""
print(spam, ham)
```

( ) Syntax Error
( ) Two empty strings
( ) An empty string and a new line character
( ) Two new line character

Answer >>>

* ord(), chr(), literals

What is the output of the following code?

```
spam = chr('a')
ham = ord(spam)
print(spam, ham)
```

( ) 97 a
( ) TypeError: an integer is required (got type str)
( ) TypeError: chr() takes exactly two arguments (1 given)
( ) TypeError: ord() takes exactly two arguments (1 given)
( ) Syntax Error

Answer >>>

* ord(), chr(), literals

What is the output of the following code?

```
"spam"
"ham"
"eggs"
print("Hello World")
```

( ) spam ham eggs Hello World
( ) Hello World
( ) SyntaxError: invalid syntax
( ) NameError: name 'spam' is not defined

Answer >>>

Which option will print the following output?

```
John said: "I'm fine!"
```

[ ] print('John said: "I\'m fine!"')
[ ] print("John said: \"I'm fine!\"")
[ ] print("John said: ""I'm fine!""")
[ ] print('John said: "I"m fine!"')
[ ] print('John said: \"I\'m fine!\"')

Answer >>>

* indexing, slicing, immutability

Which option will return a different result given the code below?

```
s = 'Python'
```

( ) print(s[0] + s[-1])
( ) print(s[::5])
( ) print(s[::-5])
( ) print(s[::-1][::-5])

Answer >>>

***** indexing, slicing, immutability

Which option will return True given the following code?

```
spam = 'FuBar'
ham = spam[:]
```

[ ] spam == ham
[ ] id(spam) == id(ham)
[ ] spam.startswith(ham)
[ ] spam.endswith(ham)
[ ] spam.equals(ham)

Answer >>>

Which option(s) will return **Ham**

[ ] 'Spam,Ham,Eggs'[5:8]
[ ] 'Spam,Ham,Eggs'[-8:-5]
[ ] 'Spam,Ham,Eggs'[5:-5]
[ ] 'Spam,Ham,Eggs'[-5:-8]
[ ] 'Spam,Ham,Eggs'[-5:5]

Answer >>>

***** indexing, slicing, immutability

What is the output of the following code?

```
spam = 'spam'
print(spam[0], end=' ')
spam[0]='x'
print(spam)
```

( ) No output
( ) s xpam

( ) s spam
( ) s followed by TypeError: 'str' object does not support item assignment

Answer >>>

What is the output of the following code?

```
>>> s = 'Hello World'
>>> for i in len(s):
...     s[i] = s[i].upper()
>>> s
```

( ) Hello World
( ) HELLO WORLD
( ) TypeError: 'str' object does not support item assignment
( ) TypeError: 'int' object is not iterable

Answer >>>

* iterating through,

What is the output of the following code?

```
s = '0123456789'
print(s[::2], s[:-2:2], s[2::2])
```

( ) 01 89 23
( ) 01 01 23
( ) 02468 0246 2468
( ) 02468 8 0
( ) SyntaxError: invalid syntax

Answer >>>

* concatenating, multiplying, comparing (against strings and numbers)

What is the output of the following code if the user enters 1 on the first prompt and 2 on the second prompt?

```
a = input("Enter first number:")
b = input("Enter second number:")
print(a + b)
```

( ) TypeError: input() takes 0 positional arguments but 1 was given

( ) 3
( ) 12
( ) TypeError: unsupported operand type(s) for +: 'str' and 'str'

Answer >>>

What is the output of the following code?

```
foo = [
    'Spam',
    'Ham'
    'Eggs'
]
print(foo)
```

( ) ['Spam', 'Ham', 'Eggs']
( ) ['Spam', 'HamEggs']
( ) ['Spam']
( ) SyntaxError: invalid syntax

Answer >>>

***** concatenating, multiplying, comparing (against strings and numbers)

What is the output of the following code?

```
>>> None * 2
```

( ) 0
( ) None
( ) NoneNone
( ) TypeError: unsupported operand type(s) for *

Answer >>>

What is the output of the following code?

```
>>> spam, ham = 1, "ham"
>>> spam *= 3
>>> ham *= 3
>>> spam, ham
```

( ) SyntaxError: invalid syntax

( ) (3, hamhamham)
( ) (3, 0)
( ) TypeError: unsupported operand type(s) for *=: 'str' and 'int'

Answer >>>

What is the output of the following code?

```
>>> 2 * 'DUN-' + 'DUUUUN!!!'
```

( ) SyntaxError: invalid syntax
( ) DUN-DUN-DUUUUN!!!
( ) 2
( ) TypeError: unsupported operand type(s) for * 'int' and 'str'

Answer >>>

What is the output of the following code?

```
>>> 2 * ('Yes' + 3*'!')
```

( ) 0
( ) 8
( ) SyntaxError: invalid syntax
( ) TypeError: unsupported operand type(s) for *: 'int' and 'str'
( ) 'Yes!!!Yes!!!'

Answer >>>

***** concatenating, multiplying, comparing (against strings and numbers)

What is the output of the following code?

```
>>> sorted([5, "1", 100, "34"])
```

( ) ["1", 5, "34", 100]
( ) [5, "1", "34", 100]
( ) ["1", "100", "34", "5"]
( ) [1, 5, 34, 100]
( ) TypeError: '<' not supported between instances of 'str' and 'int'

Answer >>>

What is the output of the following code?

```
x = "0"
y = "1"
```

```
z = "2"
x = y < z
print(x == 1, type(x))
```

( ) False <class 'bool'>
( ) True <class 'bool'>
( ) False <class 'str'>
( ) True <class 'str'>

Answer >>>

***** in, not in

What is the output of the following code?
spam.txt

```
spam ham eggs
```

spam.py

```
f = open('spam.txt', 'r')
if 'eggs' in f:
    print('Eggs found')
else:
    print('Eggs not found')
```

( ) Eggs found
( ) Eggs not found
( ) TypeError: argument type TextIOWrapper not iterable
( ) SyntaxError: invalid syntax

Answer >>>

***** in, not in

Which of the option(s) below are valid given the following code?

```
>>> '' not in 'spam'
```

[ ] Prints True
[ ] Empty string is not in the string 'spam'
[ ] Prints False
[ ] Empty string is always part of any string no exception

Answer >>>

* .isxxx(), .join(), .split()

Which of the calls below are valid String function calls and will return True?

[ ] 'abc123'.isalnum()
[ ] 'abc'.isalpha()
[ ] '123abc'.isidentifier()
[ ] '123abc'.islower()
[ ] '123'.isdigit()
[ ] 'Abc'.istitle()

Answer >>>

* .isxxx(), .join(), .split()

What is the output of the following code?

```
>>> "/".join({"Month": "12", "Day": "25", "Year":"2021"})
```

( ) 12/25/2021
( ) Month/Day/Year
( ) Month/12/Day/25/Year/2021
( ) TypeError: can only join an iterable

Answer >>>

What is the output of the following code?

```
>>> "XYZ".join("123")
```

( ) XYZ123
( ) 123XYZ
( ) 1XYZ2XYZ3
( ) X123Y123Z
( ) TypeError: can only join an iterable

Answer >>>

* .isxxx(), .join(), .split()

What is the output of the following code?

```
>>> "/spam/ham/eggs/".split("/")
```

( ) ['spam', 'ham', 'eggs']
( ) ['', 'spam', 'ham', 'eggs', '']
( ) ('spam', 'ham', 'eggs')
( ) ('', 'spam', 'ham', 'eggs', '')

Answer >>>

***** .sort(), sorted(), .index(), .find(), .rfind()

What is the output of the following code?

```
>>> spam = [4*(3+5), 4*3+5, 4+3*5, (4+3)*5]
>>> spam.sort(reverse=True)
>>> spam
```

( ) TypeError: 'reverse' is an invalid keyword argument for sort()
( ) [32, 17, 19, 35]
( ) [17, 19, 32, 35]
( ) [35, 32, 19, 17]
( ) [35, 19, 17, 32]

Answer >>>

***** .sort(), sorted(), .index(), .find(), .rfind()

What is the output of the following code?

```
d = { 'zero':0, 'one':1, 'three':3, 'two':2 }
for k in sorted(d.keys()):
    print(d[k], end=' ')
```

( ) TypeError: sorted expected 2 arguments, got 1
( ) 0 1 2 3
( ) 1 3 2 0
( ) zero one two three
( ) one three two zero

Answer >>>

What is the output of the following code?

```
>>> sorted(['banana', 'pear', 'grapes', 'apple'], key=lambda x:
x[::-1])
```

( ) ['apple', 'banana', 'grapes', 'pear']
( ) ['banana', 'apple', 'pear', 'grapes']
( ) SyntaxError: invalid syntax
( ) TypeError: 'key' is an invalid keyword argument for sorted()

Answer >>>

What is the output of the following code?

```
def reverse(word):
    return word[::-1]
print(sorted(['banana', 'pear', 'grapes', 'apple'], key=reverse))
```

( ) ['pear', 'grapes', 'banana', 'apple']
( ) ['banana', 'apple', 'pear', 'grapes']
( ) ['grapes', 'pear', 'apple', 'banana']
( ) TypeError: 'key' is an invalid keyword argument for sorted()

Answer >>>

What is the output of the following code?

```
def reverse(word):
    return word[::-1]
print(sorted(['banana', 'pear', 'grapes', 'apple'],
key=reverse, reverse=True))
```

( ) ['apple', 'banana', 'grapes', 'pear']
( ) ['grapes', 'pear', 'apple', 'banana']
( ) ['banana', 'apple', 'pear', 'grapes']
( ) TypeError: 'key' is an invalid keyword argument for sorted()

Answer >>>

* .sort(), sorted(), .index(), .find(), .rfind()

What is the output of the following code?

```
>>> "Spam Ham Eggs".index('Spam', 1)
```

( ) Spam
( ) 0
( ) 1
( ) ValueError: substring not found
( ) TypeError: index() takes 1 argument (2 given)

Answer >>>

* .sort(), sorted(), .index(), .find(), .rfind()

What is the output of the following code?

```
t = "Spam Ham"
```

```
print(t.find("Ham", 0) == t.index("Ham", 0))
print(t.find("Eggs", 0) == t.index("Eggs", 0))
```

( ) True True
( ) True False
( ) True will be printed followed by ValueError: substring not found
( ) True will be printed followed by TypeError: find() takes 1 argument (2 given)

Answer >>>

***** .sort(), sorted(), .index(), .find(), .rfind()

What is the output of the following code?

```
t = "Spam Ham"
print(t.rfind("am") == t.find("am"))
print(t.rfind("am", 3) == t.find("am", 3))
print(t.rfind("am", -3) == t.find("am", -3))
```

( ) False False False
( ) False True True
( ) True True True
( ) True will be printed followed by TypeError: rfind takes 1 argument (2 given)

Answer >>>

Exam block #4: Object-Oriented Programming (34%)

Objectives covered by the block (12 items)

* ideas: class, object, property, method, encapsulation, inheritance, grammar vs class, superclass, subclass

Which of the option(s) is valid given the code below?

```
class Spam:
    ''' This is class Spam '''
    pass
```

( ) The code compiles but will not output anything
( ) This is class Spam will be printed
( ) SyntaxError: invalid syntax
( ) The file should be saved as Spam.py

Answer >>>

What is the output of the following code?

```
def spam():
    class Ham:
        def eggs(self):
            print('Hello World')
    return Ham()

spam().eggs()
```

( ) No output
( ) Hello World
( ) SyntaxError: invalid syntax
( ) AttributeError: spam() has no attribute 'eggs'

Answer >>>

What is the output of the following code?

```
def spam():
```

```
    h = Ham()
    h.eggs()
    class Ham:
        def eggs(self):
            print('Hello World')
    return

spam()
```

( ) No output
( ) Hello World
( ) SyntaxError: invalid syntax
( ) UnboundLocalError: local variable 'Ham' referenced before assignment

Answer >>>

***** ideas: class, object, property, method, encapsulation, inheritance, grammar vs class, superclass, subclass

What is the output of the following code?

```
class Foo:
    bar = 'spam'

f1 = Foo()
f2 = Foo()
f2.bar = 'ham'
Foo.bar = 'eggs'
print(f1.bar, f2.bar, Foo.bar)
```

( ) spam ham eggs
( ) eggs ham eggs
( ) eggs eggs eggs

Answer >>>

***** ideas: class, object, property, method, encapsulation, inheritance, grammar vs class, superclass, subclass

What is the output of the following code?

```
class Spam:
    HAM = 1
    def __init__(self, v=2):
        self.v = v + Spam.HAM
        Spam.HAM += 1
a = Spam()
b = Spam(3)
print(a.v, b.v)
```

( ) TypeError: __init__() missing 1 required positional argument: 'v'
( ) 3 3
( ) 3 4
( ) 3 5

Answer >>>

What is the output of the following code?

```
class Ham:
    def __init__(self):
        self.v1 = 1
class Spam(Ham):
    def __init__(self):
        self.v2 = 2
s = Spam()
print(s.v1,s.v2)
```

( ) 0 2
( ) 1 2
( ) Invalid Syntax
( ) AttributeError: 'Spam' object has no attribute 'v1'

Answer >>>

***** ideas: class, object, property, method, encapsulation, inheritance, grammar vs class, superclass, subclass

What is the output of the following code?

```
class Ham:
```

```
    v = 1
    def v0(self):
        return self.v
class Spam(Ham):
    v = 2
s = Spam()
h = Ham()
print(s.v0(), h.v0())
```

( ) 1 1
( ) 2 1
( ) 2 2
( ) AttributeError: 'Spam' object has no attribute 'v0'

Answer >>>

What is the output of the following code?

```
>>> def foo(self, p):
...     print('Hello',p)
>>> class Spam:
...     bar = foo
>>> s = Spam()
>>> s.bar('World')
```

( ) No output
( ) Hello World
( ) SyntaxError: invalid syntax
( ) NameError: name 'foo' is not defined

Answer >>>

* ideas: class, object, property, method, encapsulation, inheritance, grammar vs class, superclass, subclass

What is the output of the following code?

```
1 class Spam:
2     def __init__(self, v):
3         self.ham = v
```

```
4        self.__ ham = self. ham + 1
5 s = Spam(100)
6 print(s.ham ,s.__ham)
```

( ) 100 101
( ) Error in Line 3
( ) Error in Line 4
( ) Error in Line 5
( ) Error in Line 6

Answer >>>

***** ideas: class, object, property, method, encapsulation, inheritance, grammar vs class, superclass, subclass

What is the output of the following code?

```
class A:
    def spam(self):
        return 'A.spam'
    def ham(self):
        return self.spam()
class B:
    def spam(self):
        return 'B.spam'
class C(B, A):
    pass

c = C()
print(c.spam(), c.ham())
```

( ) TypeError: Cannot create a consistent method resolution
( ) B.spam B.spam
( ) B.spam A.spam
( ) A.spam A.spam

Answer >>>

***** ideas: class, object, property, method, encapsulation,

inheritance, grammar vs class, superclass, subclass
  * ideas: class, object, property, method, encapsulation, inheritance, grammar vs class, superclass, subclass

What is the output of the following code?

```
class Spam:
    def foo(self):
        print('Super Spam')
class Ham:
    def foo(self):
        print('Super Ham')
class Eggs(Spam, Ham):
    def foo(self):
        super().foo()
e = Eggs()
e.foo()
```

( ) No output
( ) Super Spam
( ) Super Ham
( ) Super Spam Super Ham
( ) Super Ham Super Spam

Answer >>>

Which option(s) are valid replacements for the marked section below.

```
class Bar:
    def __init__(self):
        self.x = 1
class Foo(Bar):
    def __init__(self):
        <<< INSERT CODE HERE >>>
        self.y = 2
f = Foo()
print(f.x,f.y)
```

[ ] Blank. Code will work without replacement
[ ] super(Spam, self).__init__ ()
[ ] Bar.__init__ (self)
[ ] None. All results in AttributeError: 'Foo' object has no attribute 'x'

Answer >>>

***** ideas: class, object, property, method, encapsulation, inheritance, grammar vs class, superclass, subclass

Select the choices which will return TRUE?

```
class X:
    pass
class Y:
    pass
class Z(X, Y):
    pass
```

[ ] issubclass(X, Z) and issubclass(Y, Z)
[ ] issubclass(Z, X) and issubclass(Z, Y)
[ ] issubclass(Z, (list, X, Y))
[ ] issubclass(Z, X, Y))

Answer >>>

What is the output of the following code?

```
class A(object): pass
class C(A,A): pass
```

( ) No output
( ) SyntaxError: invalid syntax
( ) TypeError: duplicate base class A
( ) NameError: name 'object' is not defined

Answer >>>

***** instance vs class variables: declaring, initializing

What is the output of the following code?

```
class MyClass:
    FOO = 100
```

```
    def __init__(self):
        self.bar = []
    def add(self, p):
        self.bar.append(p)

d, e = MyClass(), MyClass()
d.add('spam')
e.add('ham')
e.FOO = 200
MyClass.FOO = 300
print(d.bar, d.FOO, e.bar, e.FOO)
```

( ) ['spam'] 300 ['ham'] 300
( ) ['spam'] 300 ['ham'] 200
( ) ['spam'] 100 ['ham'] 200
( ) ['spam'] 100 ['ham'] 300

Answer >>>

Which of the following option(s) is valid given the code below?

```
1 class Spam:
2     HAM = 100
3     def __init__(self):
4         self.eggs = []
5     def add(self, p):
6         self.eggs.append(p)
```

[ ] HAM is an instance variable
[ ] eggs is an instance variable
[ ] HAM is a class variable
[ ] eggs is a class variable
[ ] Error in LINE 4

Answer >>>

***** instance vs class variables: declaring, initializing

What is the output of the following code?

```
1 class Spam:
2     ham = 0
3     def __init__(self):
4         ham = 100
5
6 s, t = Spam(), Spam()
7 s.ham, t.ham = 200, 300
8 Spam.ham = 500
9 print(s.ham, t.ham)
```

( ) 500 500
( ) 200 300
( ) Error in Line 2
( ) Error in Line 4
( ) Error in Line 8

Answer >>>

***** __dict__ property (objects vs classes)

Select the option(s) which will return the dictionary or other mapping object used to store an object's (writable) attributes of the following code

```
class Person:
    name = "John"
    age = 36
    country = "USA"
p = Person()
```

[ ] vars(Person)
[ ] vars(p)
[ ] Person.__dict__
[ ] p.__dict__

Answer >>>

***** private components (instance vs classes), name mangling

What is the output of the following code?

```
class Ham:
```

```
    def __init__(self):
        print(type(self).__name__ + '.__init__()', end=' ')
        self.__update()
    def update(self):
        print(type(self).__name__ + '.update()')
    __update = update

Ham()
```

( ) The script will run but will not output anything
( ) Ham.__init__()
( ) Ham.__init__() Ham.update()
( ) AttributeError: 'Ham' object has no attribute '_Ham__update'

Answer >>>

***** private components (instance vs classes), name mangling

Which of the option(s) below are valid calls given the code below?

```
>>> class Spam:
...     __ham = 0
...     def __eggs(self):
...         __ham = 100
...         return __ham
...     eggs = __eggs
>>> s = Spam()
<<< INSERT CODE HERE >>>
```

[ ] >>> s.eggs()
[ ] >>> s.__eggs()
[ ] >>> s._Spam__eggs()
[ ] >>> s.__ham
[ ] >>> s._Spam__ham

Answer >>>

***** methods: declaring, using, self parameter

What is the output of the following code?

```
class Ham:
    def __init__(self):
        print(type(self).__name__ + '.__init__()', end=' ')
        self.update()
    def update(self):
        print(type(self).__name__ + '.update()', end=' ')
    def update(self, param):
        print(type(self).__name__ + '.update(param)', end=' ')
Ham()
```

( ) Ham.__init__() Ham.update()
( ) Ham.__init__() Ham.update(param)
( ) SyntaxError: invalid syntax
( ) TypeError: update() missing 1 required positional argument: 'param'

Answer >>>

***** methods: declaring, using, self parameter

What is the output of the following code?

```
class Ham:
    def __init__(self):
        print(type(self).__name__ + '.__init__()', end=' ')
        self.__update()
    def update(self):
        print(type(self).__name__ + '.update()', end=' ')
    __update = update

class Spam(Ham):
    def update(self, param):
        print(type(self).__name__ + '.update(param)', end=' ')

Ham()
Spam()
```

( ) Ham.__init__() Ham.update() Ham.__init__() Ham.update()

( ) Ham.__init__() Ham.update() Spam.__init__() Spam.update()
( ) Ham.__init__() Ham.update() Spam.__init__() Spam.update(param)
( ) TypeError: update() missing 1 required positional argument: 'param'

Answer >>>

What is the output of the following code?

```
class Ham:
    def __init__(self):
        print(type(self).__name__ + '.__init__()', end=' ')
        self.update()
    def update(self):
        print(type(self).__name__ + '.update()', end=' ')

class Spam(Ham):
    def update(self, param):
        print(type(self).__name__ + '.update(param)', end=' ')

Ham()
Spam()
```

( ) Ham.__init__() Ham.update() Ham.__init__() Ham.update()
( ) Ham.__init__() Ham.update() Spam.__init__() Spam.update()
( ) Ham.__init__() Ham.update() Spam.__init__() Spam.update(param)
( ) TypeError: update() missing 1 required positional argument: 'param'

Answer >>>

***** methods: declaring, using, self parameter

Which of the statements below is valid?

```
class A:
    def __init__(self):
        pass
    def spam(self):
        pass
```

```
    def ham(self):
        return <CALL spam>
a = A()
<CALL ham>
```

[ ] Replace **<CALL spam>** with self.spam()
[ ] Replace **<CALL spam>** with self.spam(self)
[ ] Replace **<CALL ham>** with a.ham()
[ ] Replace **<CALL ham>** with a.ham(a)

Answer >>>

***** introspection: hasattr() (objects vs classes), __name__, __module__, __bases__ properties

What is the output of the following code?

```
>>> class Spam: pass
>>> hasattr(Spam(), 'ham')
```

( ) False
( ) AttributeError: 'Spam' object has no attribute 'ham'
( ) TypeError: hasattr(): attribute must be type
( ) NameError: name 'hasattr' is not defined
( ) SyntaxError: invalid syntax

Answer >>>

Select the choices which will return TRUE?

```
class Spam:
    ham = 36
spam = Spam()
```

[ ] hasattr(spam, 'ham')
[ ] hasattr(Spam, 'ham')
[ ] hasattr('Spam', 'ham')
[ ] hasattr('spam', 'ham')
[ ] spam.hasattr('ham')

Answer >>>

***** introspection: hasattr() (objects vs classes), __name__, __module__, __bases__ properties

Which of the option(s) below is/are valid given the following code?

```
class Spam:
    __ham, ham = '__ham', 'ham'
    def __eggs(self):
        pass
    eggs = __eggs

s = Spam()
<<< INSERT CODE HERE >>>
```

[ ] print(Spam.__name__)
[ ] print(s.__name__)
[ ] print(s._Spam__eggs.__name__)
[ ] print(s._Spam__ham.__name__)
[ ] print(s.eggs.__name__)
[ ] print(s.ham.__name__)

Answer >>>

***** introspection: hasattr() (objects vs classes), __name__, __module__, __bases__ properties

Which of the option(s) below are valid given the following code?

```
class Spam:
    __ham, ham = '__ham', 'ham'
    def __eggs(self):
        pass
    eggs = __eggs

s = Spam()
<<< INSERT CODE HERE >>>
```

[ ] print(__module__)
[ ] print(Spam.__module__)
[ ] print(s._Spam__eggs.__module__)
[ ] print(s.eggs.__module__)
[ ] print(s.__module__)

[ ] print(s.ham.__module__)

Answer >>>

***** introspection: hasattr() (objects vs classes), __name__, __module__, __bases__ properties

Which of the option(s) below are valid given the following code?

```
class Spam:
    __ham, ham = '__ham', 'ham'
    def __eggs(self):
        pass
    eggs = __eggs

s = Spam()
<<< INSERT CODE HERE >>>
```

[ ] print(__bases__)
[ ] print(Spam.__bases__)
[ ] print(s._Spam__eggs.__bases__)
[ ] print(type(s.eggs).__bases__)
[ ] print(type(s).__bases__)
[ ] print(s.ham.__bases__)

Answer >>>

***** inheritance: single, multiple, isinstance(), overriding, not is and is operators

What is the output of the following code?

```
class Eggs:
    def __init__(self):
        print('Eggs', end=' ')
class Ham(Eggs):
    def __init__(self):
        print('Ham', end=' ')
class Spam(Ham):
    pass
```

```
s = Spam()
```
( ) No output
( ) Ham
( ) Eggs
( ) Ham Eggs
( ) TypeError: __init__() takes 1 positional argument but 0 were given

Answer >>>

* inheritance: single, multiple, isinstance(), overriding, not is and is operators

What is the output of the following code?

```
class Ham:
    def __init__(self):
        print('Ham', end=' ')
class Eggs:
    def __init__(self, end=' '):
        print('Eggs')
class Spam(Ham, Eggs):
    pass
```
( ) No output
( ) Ham
( ) Eggs
( ) Ham Eggs
( ) TypeError: __init__ takes 1 positional argument but 0 were given

Answer >>>

* inheritance: single, multiple, isinstance(), overriding, not is and is operators

Select the choices which will return TRUE?

```
>>> class X: pass
>>> class Y: pass
>>> class Z(X, Y): pass
```

```
>>> x, y, z = X(), Y(), Z()
```

[ ] isinstance(X, z) and isinstance(Y, z)
[ ] isinstance(z, X) and isinstance(z, Y)
[ ] isinstance(z, (list, X, Y))
[ ] isinstance((list, X, Y), z)
[ ] isinstance(z, X, Y)

Answer >>>

Which option will return True given the following code?

```
>>> class A(object): pass
>>> class B(object): pass
>>> class C(object): pass
>>> class D(object): pass
>>> class E(object): pass
>>> class K1(A,B,C): pass
>>> class K2(D,B,E): pass
>>> class K3(D,A):   pass
>>> k = K3()
```

[ ] >>> isinstance(k, K3)
[ ] >>> isinstance(k, D)
[ ] >>> isinstance(k, (list, K2, K3))
[ ] >>> isinstance(k, (list, K1, K2))
[ ] >>> isinstance(k, (list, A, B, C, D, E))

Answer >>>

***** inheritance: single, multiple, isinstance(), overriding, not is and is operators

Select which option contains the correct function name for the following generator?

```
class Spam:
    def <<Replace 1>>(self, p=""):
        self.s = p
        self.i = 0
    def <<Replace 2>>(self):
```

```
        return self
    def <<Replace 3>>(self):
        if self.i == len(self.s):
            raise StopIteration
        v = self.s[self.i]
        self.i += 1
        return v
```

( ) 1=__init__, 2=__iter__, 3=__next__
( ) 1=__init__, 2=__iterator__, 3=__next__
( ) 1=__init__, 2=__iterate__, 3=__next__
( ) 1=__init__, 2=__pop__, 3=__push__
( ) 1=__init__, 2=__generator__, 3=__next

Answer >>>

* inheritance: single, multiple, isinstance(), overriding, not is and is operators
* constructors: declaring and invoking

Select the choices to invoke the constructor of Spam and assign the instance to s

```
class Spam:
    def __init__(self, v=0):
        self.ham = v + 1
```

[ ] s = Spam()
[ ] s = Spam(10)
[ ] s = Spam(s, 10)
[ ] s = Spam.__init__(s)
[ ] AttributeError: 'Spam' object has no attribute 'ham'

Answer >>>

* constructors: declaring and invoking

How do you instantiate class Spam of the code below?

```
class Spam:
    def __init__(self):
```

```
        self.bar = 0
```

( ) You can't because there's an AttributeError in the code
( ) spam = Spam()
( ) spam = Spam(None)
( ) spam = Spam(Spam)

Answer >>>

What is the output of the following code?

```
class Spam:
    def __init__(self, v):
        self.ham = v + 1
spam = Spam(1)
print(spam.ham)
```

( ) AttributeError: 'Spam' object has no attribute 'ham'
( ) TypeError: __init__() takes 2 positional arguments but 1 were given
( ) 1
( ) 2

Answer >>>

* polymorphism
* __name__, __module__, __bases__ properties, __str__() method

Select the line number from the options(s) which will print **Spam**

```
1 class Spam:
2     def v0(self):
3             print(__name__)
4 print(__name__)
5 s = Spam()
6 s.v0()
7 print(s.__class__.__name__)
8 print(Spam.__name__)
9 print(s.__name__)
```

[ ] Line 3

[ ] Line 4
[ ] Line 7
[ ] Line 8
[ ] Line 9

Answer >>>

What is the output of the following code?

```
class X:
    def spam(): pass
ham = X.spam
print(ham.__name__)
```

( ) SyntaxError: invalid syntax
( ) X.spam
( ) spam
( ) ham

Answer >>>

***** __name__, __module__, __bases__ properties, __str__() method

What is the output of the following code?

```
F=type('Food',(),{'remember2buy':'spam'})
E=type('Eggs',(F,),{'remember2buy':'eggs'})
G=type('GoodFood',(E,F),{})
print(F.__name__, E.__name__, G.__name__)
```

( ) No output
( ) SyntaxError: invalid syntax
( ) F E G
( ) Food Eggs GoodFood
( ) AttributeError: type object 'Food' has no attribute '__name__'

Answer >>>

***** __name__, __module__, __bases__properties, __str__() method
***** __name__, __module__, __bases__ properties, __str__() method

* __name__, __module__, __bases__ properties, __str__() method

What is the output of the following code?

```
class Ham:
    def __str__(self): return "Ham"

class Spam(Ham): pass

print(Spam())
```

( ) No output
( ) <__main__.Spam object at 0x03100FD0>
( ) Ham
( ) TypeError: __str__() missing 1 required positional argument

Answer >>>

Which option will print **Spam** given the following code?

```
>>> class Spam:
...     def __str__(self): return "Spam"
>>> s = Spam()
```

[ ] >>> s
[ ] >>> print(s)
[ ] >>> Spam()
[ ] >>> s.__str__()
[ ] >>> s.__repr__()

Answer >>>

* multiple inheritance, diamonds

What is the output of the following code?

```
F=type('Food',(),{'remember2buy':'spam'})
E=type('Eggs',(F,),{'remember2buy':'eggs'})
G=type('GoodFood',(E,F),{})
print(F.remember2buy, E.remember2buy, G.remember2buy)
```

( ) SyntaxError: invalid syntax

( ) No Output
( ) Food Eggs GoodFood
( ) spam eggs
( ) spam eggs eggs

Answer >>>

* multiple inheritance, diamonds

Which of the option(s) below are valid given the following code?

```
O = object
class X(O): pass
class Y(O): pass
class A(X,Y): pass
class B(Y,X): pass
<<< INSERT CODE HERE >>>
```

[ ] class Foo(A, B): pass
[ ] class Foo(B, A): pass
[ ] class Foo(A, X): pass
[ ] class Foo(X, A): pass
[ ] class Foo(B, Y): pass
[ ] class Foo(Y, B): pass

Answer >>>

Which of the option(s) below are valid given the following code?

```
O = object
class F(O): pass
class E(O): pass
class D(O): pass
class C(D,F): pass
class B(D,E): pass
class A(B,C): pass
<<< INSERT CODE HERE >>>
```

[ ] class Foo(A, B): pass
[ ] class Foo(B, A): pass
[ ] class Foo(A, C): pass

[ ] class Foo(C, A): pass
[ ] class Foo(B, C): pass
[ ] class Foo(C, B): pass
[ ] class Foo(C, B, E): pass
[ ] class Foo(E, B, C): pass

Answer >>>

Which of the options below are valid given the following code?

```
class A(object): pass
class B(object): pass
class C(object): pass
class D(object): pass
class E(object): pass
class K1(A,B,C): pass
class K2(D,B,E): pass
class K3(D,A):   pass
<<< INSERT CODE HERE >>>
```

[ ] class Foo(K1,K2,K3): pass
[ ] class Foo(K1,K3,K2): pass
[ ] class Foo(K2,K1,K3): pass
[ ] class Foo(K2,K3,K1): pass
[ ] class Foo(K3,K1,K2): pass
[ ] class Foo(K3,K2,K1): pass

Answer >>>

Exam block #5: Miscellaneous (List Comprehensions, Lambdas, Closures, and I/O Operations) (22%)

Objectives covered by the block (9 items)

* list comprehension: if operator, using list comprehensions

What is the output of the following code?

```
>>> [[c for c in range(r)] for r in range(3) if r != 0]
```

( ) [[0], [0, 1]]
( ) [[1], [1, 2]]
( ) [[0], [1]]
( ) [[1], [2]]
( ) SyntaxError: invalid syntax

Answer >>>

What is the output of the following code?

```
>>> [_ for _ in range(10) if not _%2 ]
```

( ) [0, 2, 4, 6, 8]
( ) [2, 4, 6, 8, 10]
( ) [1, 3, 5, 7, 9]
( ) [0, 1, 3, 4, 5, 6, 7, 8, 9]
( ) SyntaxError: invalid syntax

Answer >>>

* list comprehension: if operator, using list comprehensions

What is the output of the following code?

```
>>> [False for i in range(3)]
```

( ) []
( ) [False]
( ) [0, 1, 2]
( ) [False, False, False]

( ) SyntaxError: invalid syntax

Answer >>>

What is the output of the following code?

```
>>> [i for i in range(1)][-1]
```

( ) SyntaxError: invalid syntax
( ) []
( ) [0]
( ) 0
( ) None

Answer >>>

What is the output of the following code?

```
>>> len([[c for c in range(r)] for r in range(3)])
```

( ) 2
( ) 3
( ) TypeError: len() takes exactly one argument (2 given)
( ) SyntaxError: invalid syntax

Answer >>>

Which option will produce a non-empty list?

[ ] lst = [i for i in range(1, 5)]
[ ] lst = [i for i in range(5, 1)]
[ ] lst = [i for i in range(-1, -5)]
[ ] lst = [i for i in range(-5, -1)]
[ ] lst = [i for i in range(0, -5)]

Answer >>>

What is the output of the following code?

```
spam = [x * x for x in range(5)]
del spam[spam[2]]
print(spam)
```

( ) [0, 1, 4, 9]
( ) IndexError: list index out of range
( ) TypeError: range() takes exactly 2 arguments (1 given)
( ) SyntaxError: invalid syntax

Answer >>>

What is the output of the following code?

```
x = [_ for _ in range(10)]
del x[0:-2]
print(x)
```

( ) [8, 9]
( ) [9, 8]
( ) [0, 1, 2, 3, 4, 5, 6, 7]
( ) [7, 6, 5, 4, 3, 2, 1, 0]
( ) [2, 3, 4, 5, 6, 7, 8, 9]
( ) [9, 8, 7, 6, 5, 4, 3, 2]

Answer >>>

What is the output of the following code?

```
>>> [i // i for i in range(0,3)]
```

( ) [0, 1, 1]
( ) [1, 1]
( ) [0, 1, 2]
( ) ZeroDivisionError: integer division or modulo by zero

Answer >>>

What is the output of the following code?

```
>>> [2 ** x for x in range(5)]
```

( ) [0, 2, 4, 6, 8]
( ) [1, 2, 4, 8, 16]
( ) [2, 4, 6, 8, 10]
( ) SyntaxError: invalid syntax

Answer >>>

What is the output of the following code?

```
spam = [[x for x in range(4)] for y in range(4)]
for r in range(4):
    for c in range(4):
        spam[r][c] += 5
```

```
print(spam)
```

( ) [[5, 6, 7, 8], [5, 6, 7, 8]]
( ) [[5, 6, 7, 8], [5, 6, 7, 8], [5, 6, 7, 8], [5, 6, 7, 8]]
( ) Invalid Syntax
( ) TypeError: 'list' object does not support item assignment

Answer >>>

How many stars will the following code print?

```
l = [[i for i in range(2)] for i in range(2)]
for i in range(2):
    if l[0][i] % l[1][i] == 0:
        print('*')
```

( ) 0
( ) 1
( ) 2
( ) 4

Answer >>>

***** lambdas: defining and using lambdas, self-defined functions taking lambda as as arguments; map(), filter();

Which option is a valid definition of a lambda assigned to f that adds the parameter x and y?

[ ] f = lambda x, y : x + y
[ ] f = lambda (x, y):(x + y)
[ ] f = lambda (x, y): x + y
[ ] f = lambda x, y : (x + y)

Answer >>>

What is the output of the following code?

```
f = lambda x: 10
print(f(20))
```

( ) 0
( ) 10
( ) 20
( ) SyntaxError: invalid syntax

Answer >>>

What is the output of the following code?

```
func = lambda x: return x
print(func(10))
```

( ) No output
( ) 10
( ) NameError: name 'x' is not defined
( ) SyntaxError: invalid syntax

Answer >>>

What is the output of the following code?

```
>>> (lambda x: assert x != 2)(2)
```

( ) No output
( ) AssertionError
( ) NameError: name 'x' is not defined
( ) SyntaxError: invalid syntax

Answer >>>

What is the output of the following code?

```
>>> a, b = 10, 20
>>> (lambda: b, lambda: a)[a < b]()
```

( ) 10
( ) 20
( ) TypeError: tuple indices must be integers or slices
( ) Invalid Syntax

Answer >>>

***** lambdas: defining and using lambdas, self-defined functions taking lambda as as arguments; map(), filter();

What is the output of the following code?

```
spam = lambda x, f: x + f(x)
print(spam(2, lambda x: x * x), end=' ')
print(spam(2, lambda x: x + 3))
```

( ) SyntaxError: invalid syntax

( ) 6 7
( ) 4 5
( ) 6 10

Answer >>>

* lambdas: defining and using lambdas, self-defined functions taking lambda as as arguments; map(), filter();

What is the output of the following code?

```
>>> list(map(lambda x: x*2, range(3)))
```

( ) [0, 1, 2]
( ) [0, 2, 4]
( ) [[0,0], [1,2], [2,4]]
( ) SyntaxError: invalid syntax

Answer >>>

What is the output of the following code?

```
>>> list(map(lambda x: x+10, [1, 2, 3]))
```

( ) [11, 12, 13]
( ) [11, 12, 13, 1, 2, 3]
( ) [[11, 12, 13],[1, 2, 3]]
( ) [{11:1},{12:2},{13:3}]
( ) [{11:[1, 2, 3]},{12:[1, 2, 3]},{13:[1, 2, 3]}]
( ) Invalid Syntax

Answer >>>

* lambdas: defining and using lambdas, self-defined functions taking lambda as as arguments; map(), filter()

What is the output of the following code?

```
>>> list(filter(lambda x: x%2, [1, 2, 3, 4, 5, 6, 7, 8, 9]))
```

( ) [1, 2, 3, 4, 5, 6, 7, 8, 9]
( ) [1, 3, 5, 7, 9]
( ) [2, 4, 6, 8]
( ) [1, 0, 1, 0, 1, 0, 1, 0, 1]
( ) [0, 1, 0, 1, 0, 1, 0, 1, 0]

Answer >>>

* closures: meaning, defining, and using closures

Which option is True about closures?

[ ] closures is always nested inside a function
[ ] closures can be defined outside a function
[ ] closures have access to a free variable in outer scope
[ ] closures have no access to variables in outer scope
[ ] closures have access to global variables
[ ] closures can define and modify nonlocal variables
[ ] closures are not allowed to define nonlocal variables

Answer >>>

* closures: meaning, defining, and using closures

What is the output of the following code?

```
x = 'x'
def main():
    y = 'y'
    def spam():
        print(x, y, z)
        return
    spam()
z = 'z'
if __name__ == "__main__":
    main()
```

( ) No output
( ) x y z
( ) NameError: name 'z' is not defined
( ) SyntaxError: invalid syntax

Answer >>>

* closures: meaning, defining, and using closures

What is the output of the following code?

```
def spam(x):
    def ham(y):
```

```
        return x * y
    return ham
f = spam(2)
print(f(3))
```

( ) SyntaxError: invalid syntax
( ) 0
( ) 6
( ) NameError: name 'y' is not defined

Answer >>>

What is the output of the following code?

```
def spam(x):
    y = 2
    def ham(z):
        return x + y + z
    return ham

for i in range(3):
    eggs = spam(i)
    print(eggs(i+3), end=' ')
```

( ) 5 7 9
( ) 6 8 10
( ) 5 6 7
( ) SyntaxError: invalid syntax
( ) NameError: name z is not defined

Answer >>>

What is the output of the following code?

```
def spam(x):
    y = 2
    return lambda z: x + y + z

for i in range(3):
    eggs = spam(i)
```

```
    print(eggs(i+3), end=' ')
```

( ) 5 7 9
( ) 6 8 10
( ) 5 6 7
( ) SyntaxError: invalid syntax
( ) NameError: name z is not defined

Answer >>>

What is the output of the following code?

```
def main():
    x = 100
    a = [x, 200, 300]
    def spam():
        x = 500
        a[0] = x
        return
    spam()
    print(x, a)
if __name__ == "__main__":
    main()
```

( ) 100 [100, 200, 300]
( ) 100 [500, 200, 300]
( ) 500 [500, 200, 300]
( ) SyntaxError: invalid syntax
( ) NameError: name x is not defined

Answer >>>

***** I/O Operations: I/O modes, predefined streams, handles; text/binary modes
open(), errno and its values; close()
.read(), .write(), .readline(); readlines() (along with bytearray())

Select the choice(s) which is TRUE?

```
1 import sys
2 temp = sys.stdout
```

```
3 sys.stdout = open('spam.txt', 'w')
4 print("Hello World")
5 sys.stdout.close()
6 sys.stdout = temp
7 print("Good Bye")
```

[ ] An empty file 'spam.txt' will be created and the screen will display the text "Hello World" and "Good Bye"
[ ] A file 'spam.txt' containing "Hello World" will be created and the screen will display the text "Good Bye"
[ ] A file 'spam.txt' containing "Hello World" will be created and the screen will display the text "Hello World" and "Good Bye"
[ ] No file will be created and the screen will display the text "Hello World" and "Good Bye"
[ ] io.UnsupportedOperation in line 3

Answer >>>

What will open("spam.txt", "rt") return?

( ) Numeric status code
( ) The entire content of 'spam.txt'
( ) String filename
( ) File object
( ) TypeError: open() argument 2 must be int, not str

Answer >>>

***** I/O Operations: I/O modes, predefined streams, handles; text/binary modes
open(), errno and its values; close()
.read(), .write(), .readline(); readlines() (along with bytearray())

How do you use the BytesIO class in the io package imported on the following code?

```
import io
```

( ) spam = BytesIO()
( ) spam = io.BytesIO()
( ) spam = io->BytesIO()
( ) spam = io:BytesIO()

( ) spam = io::BytesIO()

Answer >>>

What functions can you call to read the Buffered Stream on the following code?

```
import io
spam = io.BytesIO()
spam.write("Hello, world!".encode('ascii'))
ham = spam.getbuffer()
spam.seek(0)
```

[ ] spam.read()
[ ] spam.read1()
[ ] spam.readinto(ham)
[ ] spam.readinto1(ham)
[ ] spam.read(ham)

Answer >>>

Choose all correct ways to create a bytearray bar?

[ ] bar = b'confuse the cat'
[ ] bar = bytearray()
[ ] bar = bytearray(range(10))
[ ] bar = bytearray('confuse the cat')
[ ] bar = bytearray(b'confuse the cat')

Answer >>>

***** I/O Operations: I/O modes, predefined streams, handles; text/binary modes
open(), errno and its values; close()
.read(), .write(), .readline(); readlines() (along with bytearray())

***** I/O Operations: I/O modes, predefined streams, handles; text/binary modes
open(), errno and its values; close()
.read(), .write(), .readline(); readlines() (along with bytearray())

What is the output of the code?

```
for spam in open('spam.txt', 'rt'):
```

```
    print(spam, end='')
```

given spam.txt

```
This is LINE 1
This is LINE 2
```

( ) Nothing is printed
( ) This is LINE 1 is printed in an infinite loop
( ) This is LINE 1 This is LINE 2 is printed in a single line
( ) This is LINE 1 and This is LINE 2 is printed in separate lines
( ) TypeError: 'TextIOWrapper' object is not iterable

Answer >>>

What are the valid access modes available for the open() function?

[ ] r, w, a
[ ] rb, wb, ab
[ ] r+, w+, a+
[ ] rb+, wb+, ab+
[ ] br, bw, ba
[ ] br+, bw+, ba+
[ ] r+b, w+b, a+b

Answer >>>

***** I/O Operations: I/O modes, predefined streams, handles; text/binary modes
open(), errno and its values; close()
.read(), .write(), .readline(); readlines() (along with bytearray())

Which option(s) are valid results of the following code if spam.txt does not exist?

```
import sys, errno
try:
    open("spam.txt", "r")
except:
    if sys.exc_info()[1].errno == errno.ENOENT:
        print(errno.errorcode[sys.exc_info()[1].errno])
```

[ ] No output

[ ] 2
[ ] ENOENT
[ ] No such file or directory
[ ] Error Line 3
[ ] Error Line 5
[ ] Error Line 6

Answer >>>

Which option(s) are valid results of the following code if spam.txt does not exist?

```
import sys, errno
try:
    open("spam.txt", "x")
except:
    if sys.exc_info()[1].errno == errno.ENOENT:
        print(errno.errorcode[sys.exc_info()[1].errno])
```

[ ] No output
[ ] 2
[ ] ENOENT
[ ] No such file or directory
[ ] Error Line 3
[ ] Error Line 5
[ ] Error Line 6

Answer >>>

Which option(s) are valid results of the following code if spam.txt exist?

```
import sys
try:
    open("spam.txt", "x")
except:
    print(sys.exc_info()[1].errno)
```

[ ] No output
[ ] invalid mode: 'x'
[ ] [Errno 17] File exists: 'spam.txt'
[ ] Error Line 3

[ ] Error Line 5

Answer >>>

***** I/O Operations: I/O modes, predefined streams, handles; text/binary modes
open(), errno and its values; close()
.read(), .write(), .readline(); readlines() (along with bytearray())

Which of the option(s) below are valid calls given the code below?

```
file = open('spam.txt', 'r+')
file.close()
<<< INSERT CODE HERE >>>
```

[ ] No file operations are allowed after close()
[ ] file.close()
[ ] file.read()
[ ] file.readline()
[ ] file.write()

Answer >>>

***** I/O Operations: I/O modes, predefined streams, handles; text/binary modes
open(), errno and its values; close()
.read(), .write(), .readline(); readlines() (along with bytearray())

What is the output of the code?

```
spam = open("spam.txt", "r")
print(spam.read(2))
```

given spam.txt

```
This is LINE 1
This is LINE 2
This is LINE 3
```

( ) The first 2 characters **Th** is printed
( ) The first 2 lines **This is LINE 1** and **This is LINE 2** is printed
( ) The first 2 characters **Th** is skipped **is is LINE 1** is printed
( ) The 2nd line **This is LINE 2** is printed

Answer >>>

***** I/O Operations: I/O modes, predefined streams, handles; text/binary modes
open(), errno and its values; close()
.read(), .write(), .readline(); readlines() (along with bytearray())

Which option(s) will write "Hello World" on an open file f?

[ ] f.write("Hello World")
[ ] f.writeln("Hello World")
[ ] f.writeline("Hello World")
[ ] f.writelines("Hello World")
[ ] f.writelines(["Hello World"])
[ ] f.print("Hello World")

Answer >>>

What is the result of the following code?

```
with open('spam.txt', 'r+') as file:
    line = file.read()
file.write(line)
```

given spam.txt

```
12345
```

[ ] spam.txt will still contain 12345
[ ] spam.txt will now contain 123451
[ ] spam.txt will now contain 1234512345
[ ] NameError: name 'line' is not defined
[ ] ValueError: I/O operation on closed file.

Answer >>>

***** I/O Operations: I/O modes, predefined streams, handles; text/binary modes
open(), errno and its values; close()
.read(), .write(), .readline(); readlines() (along with bytearray())

Which option(s) are valid read on an open file f?

[ ] f.read()
[ ] f.read(1)

[ ] f.readln()
[ ] f.readline()
[ ] f.readlines()
[ ] f.readlines(1)

Answer >>>

What is the output of the code?

```
spam = open("spam.txt", "r")
print(spam.readline(2))
```

given spam.txt

```
This is LINE 1
This is LINE 2
This is LINE 3
```

( ) The first 2 characters **Th** is printed
( ) The first 2 lines **This is LINE 1** and **This is LINE 2** is printed
( ) The first 2 characters **Th** is skipped **is is LINE 1** is printed
( ) The 2nd line **This is LINE 2** is printed

Answer >>>

***** I/O Operations: I/O modes, predefined streams, handles; text/binary modes
open(), errno and its values; close()
.read(), .write(), .readline(); readlines() (along with bytearray())

What is the output of the following code?

```
>>> b'the quick brown fox'.translate(None, b'aeiou')
```

( ) b'aeiou'
( ) b'th qck brwn fx'
( ) aeiou
( ) th qck brwn fx
( ) the quick brown fox

Answer >>>

Exam block #6: Bonus questions

Python 3.9.4 REPL can be launched by which of the following option given the output of py -0?

```
C:\Users\jlacanienta>py -0
Installed Pythons found by py Launcher for Windows
-3.9-64 *
```

[ ] py
[ ] py -3
[ ] py -3.9
[ ] py -3.9-64
[ ] py -3.9.4
[ ] py -3.9.4-64

Answer >>>

What is the output of the following code?

```
a, b = 10, 20
print(a < b and a or b)
```

( ) 10
( ) 20
( ) True
( ) Invalid Syntax

Answer >>>

Select all valid variable names

[ ] is
[ ] then
[ ] elif
[ ] pass
[ ] catch
[ ] exception

Answer >>>

Select the values considered false.

[ ] None

[ ] False
[ ] zero of any numeric type (0, 0L, 0.0, 0j)
[ ] empty sequence ('', (), [])
[ ] class with a __nonzero__()definition
[ ] class with __len__() that returns integer 0

Answer >>>

Select all valid bitwise operators

[ ] <<
[ ] >>
[ ] &
[ ] |
[ ] ~
[ ] ^

Answer >>>

Which option(s) results in 12.34?

[ ] >>> 1234e2
[ ] >>> 1234e-2
[ ] >>> .1234e2
[ ] >>> .1234e-2
[ ] None

Answer >>>

Select valid integer assignment for the variable spam?

[ ] spam = 1e0
[ ] spam = 0b1
[ ] spam = 0o1
[ ] spam = 0x1
[ ] spam = \u0031 #The Unicode of 1 is U+0031

Answer >>>

What is the output of the following code?

```
>>> ['spam', 'ham'] * 2
```

( ) SyntaxError: invalid syntax
( ) TypeError: unsupported operand type(s) for *: 'list' and 'int'
( ) ['spamspam', 'hamham']

( ) ['spam', 'ham', 'spam', 'ham']

Answer >>>

What is the output of the following code?

```
>>> ('spam', ) * 2
```

( ) SyntaxError: invalid syntax
( ) TypeError: unsupported operand type(s) for *: 'tuple' and 'int'
( ) 'spamspam'
( ) ('spamspam')
( ) ('spam', 'spam')

Answer >>>

What is the output of the following code?

```
>>> 1 - 2 / 3 // 4 + 5
```

( ) 4.0
( ) 5.0
( ) 6.0
( ) 7.0

Answer >>>

What is the output of the following code?

```
>>> 1 // 2 + 1 / 2
```

( ) 0.0
( ) 0.5
( ) 0.75
( ) 1.0

Answer >>>

What is the output of the following code?

```
>>> 1. /( 4. % 2.)
```

( ) 0.5
( ) 0.0
( ) Syntax Error
( ) ZeroDivisionError

Answer >>>

What is the output of the following code?

```
x = 3
while x > 0:
    print(x, end='')
    x //= 2
```

( ) 3
( ) 31
( ) 31.50
( ) Infinite loop

Answer >>>

What is the output of the following code?

```
>>> -1 // 2
```

( ) -1
( ) -1.0
( ) -0.5
( ) 0
( ) 0.5
( ) 1.0
( ) 1

Answer >>>

What is the output of the following code?

```
a, b = 0, 1
print(a ^ a, a ^ b, b ^ a, b ^ b)
```

( ) 0 1 1 1
( ) 0 1 1 0
( ) 1 0 0 1
( ) 1 0 0 0

Answer >>>

What is the output of the following code?

```
t = True
f = not t
```

```
t = t or f
f = t and f
t, f = f, t
print(t, f)
```

( ) False False
( ) False True
( ) True False
( ) True True
( ) Syntax Error

Answer >>>

Select options which will print True based on the following code?

```
spam = True
```

[ ] print(spam = True)
[ ] print(spam == True)
[ ] print(spam === True)
[ ] print(spam is True)

Answer >>>

Select options which will print True based on the following code?

```
spam = 1
```

[ ] print(spam != 0)
[ ] print(spam !== 0)
[ ] print(spam !=== 0)
[ ] print(spam <> 0)
[ ] print(spam is not 0)

Answer >>>

What is the output of the following code?

```
a = 10
b = 20
c = b < a > 0 or a > b and b > a or a < b
print(c)
```

( ) SyntaxError: invalid syntax

( ) True
( ) False
( ) 1
( ) 0

Answer >>>

Select all keyword argument of print()

[ ] sep
[ ] end
[ ] file
[ ] flush
[ ] format

Answer >>>

What is the output of the following code?

```
1 print(int(10.10), end= " ")
2 print(int("10", 10), end= " ")
3 print(int("10", base=10), end= " ")
4 print(int(0o12), end= " ")
5 print(int(10))
```

( ) 10 10 10 10 10
( ) TypeError: int() takes at most 1 argument (2 given)
( ) TypeError: 'base' is an invalid keyword argument for int()
( ) Invalid syntax on Line 4 (0o12)

Answer >>>

What is the output of the code?

```
1 print(float('+1.23'), end=" ")
2 print(float(' -12345\n'), end=" ")
3 print(float('1e-003'), end=" ")
4 print(float('+1E6'), end=" ")
5 print(float('-Infinity'))
```

( ) 1.23 -12345.0 0.001 1000000.0 -inf
( ) ValueError: could not convert string to float in Line 1
( ) ValueError: could not convert string to float in Line 2

( ) ValueError: could not convert string to float in Line 3
( ) ValueError: could not convert string to float in Line 4
( ) ValueError: could not convert string to float in Line 5

Answer >>>

What will call to output x/y/z*?

```
x, y, z = "x", "y", "z"
s = [x, y, z]
t = x, y, z
```

[ ] print(x, y, z, sep='/', end="*\n")
[ ] print(s, sep='/', end="*\n")
[ ] print(t, sep='/', end="*\n")
[ ] print('/'.join(s) + '*\n')
[ ] Syntax Error

Answer >>>

What is the output of the following code?

```
>>> spam = ('S', 'P', 'A', 'M')
>>> s, p, _, _ = spam
>>> s
'S'
>>> p
'P'
>>> _
```

( ) No output
( ) 'A'
( ) 'M'
( ) 'AM'

Answer >>>

What is the output of the following code?

```
>>> for i in range(10):
...     pass
>>> i
```

( ) NameError: name 'i' is not defined
( ) 0
( ) 9
( ) 10

Answer >>>

What is the output of the following code?

```
total = 0
for i in range(1, 4) :
    i += 2
    total += i
else:
    total += 100
print(total)
```

( ) 112
( ) 12
( ) 108
( ) 8
( ) SyntaxError: invalid syntax

Answer >>>

What is the output of the following code?

```
for i in range(1, 4, 2):
    print(i, end=' ')
```

( ) TypeError: range expected at most 2 arguments, got 3
( ) 1 3
( ) 1, 4, 2
( ) 1 2 3 4

Answer >>>

What is the output of the following code?

```
x = { 'x': 1, 'y': 2}
for e in x:
    print(e, type(e), end=' ')
```

( ) x <class 'str'> y <class 'str'>
( ) 1 <class 'int'> 2 <class 'int'>
( ) ('x', 1) <class 'tuple'> ('y', 2) <class 'tuple'>
( ) 'x':1 <class 'iterable'> 'y':2 <class 'iterable'>

Answer >>>

What is the output of the following code?

```
i = 0
total = 0
while i < 4 :
    i += 2
    total += i
else:
    total += 100
print(total)
```

( ) 4
( ) 6
( ) 104
( ) 106
( ) SyntaxError: invalid syntax

Answer >>>

What is the output of the following code?

```
spam = [1, 2]
for i in range(2):
    spam.insert(-1, spam[i])
    print(spam, end=' ')
```

( ) [1, 1, 2] [1, 1, 1, 2]
( ) [1, 1, 2] [1, 1, 2, 2]
( ) [-1, 1, 2] [-1, -1, 1, 2]
( ) [1, 2, -1] [1, 2, -1, -1]
( ) [1, -1, 2] [1, -1, -1, 2]

Answer >>>

Which statement is CORRECT about the code below?

```
class Spam: pass
del Spam
ham = "Fubar"
del ham
ham = 'FuBar'
del ham[0]
ham = ["spam", "ham"]
del ham[1]
```

( ) No error
( ) del Spam is an error
( ) del ham is an error
( ) del ham[0] is an error
( ) del ham[1] is an error

Answer >>>

What is the output of the following code?

```
spam = [1, 2, 3]
ham = spam
del ham[:]
print(spam)
```

( ) [1, 2, 3]
( ) []
( ) Can't delete list
( ) SyntaxError: invalid syntax

Answer >>>

Which option will result in the output **[1, 2, 3] [1, 2, 3] False**?

```
ham = [1, 2, 3]
<<< INSERT CODE HERE >>>
print(spam, ham, id(spam)==id(ham))
```

[ ] spam = ham
[ ] spam = ham.copy()
[ ] spam = ham[:]
[ ] spam = list(ham)

Answer >>>

Which option will print "the quick brown fox" given the code below?

```
spam = ("the", "quick", "brown", "fox")
```

[ ] print(spam[0],spam[1],spam[2],spam[3])
[ ] print(spam[1],spam[2],spam[3],spam[4])
[ ] print(spam[0],spam[-1],spam[-2],spam[-3])
[ ] print(spam[-4],spam[-3],spam[-2],spam[-1])

Answer >>>

What is the output of the following code?

```
a, b = 10, 20
print((b, a) [a < b])
```

( ) 10
( ) 20
( ) TypeError: tuple indices must be integers or slices
( ) Invalid Syntax

Answer >>>

Which option will create a tuple ham equal to ("brown", "fox") using the code below?

```
spam = ("the", "quick", "brown", "fox")
```

[ ] ham = spam[2:]
[ ] ham = spam[2:4]
[ ] ham = spam[-2:]
[ ] ham = spam[-2:0]
[ ] ham = spam[-2:-1]

Answer >>>

Which option will create a tuple ham equal to (0, 3, 6, 9) using the code below?

```
spam = (0, 1, 2, 3, 4, 5, 6, 7, 8, 9)
```

[ ] ham = spam[::3]
[ ] ham = spam[0:3:9]
[ ] ham = spam[0::3]

[ ] ham = spam[0:9:3]
[ ] ham = spam[0:10:3]

Answer >>>

Which option is a valid way to create a tuple named spam?

[ ] spam = ()
[ ] spam = ("the")
[ ] spam = ("the",)
[ ] spam = ("the", "quick", "brown", "fox")

Answer >>>

What is the output of the following code?

```
1 spam = ('0', 1, 2, 3, 4, 5, 6, 7, 8, 9)
2 spam[0] = 0
3 print(spam)
```

( ) ('0', 1, 2, 3, 4, 5, 6, 7, 8, 9)
( ) (0, 1, 2, 3, 4, 5, 6, 7, 8, 9)
( ) Error in Line 1
( ) Error in Line 2
( ) Error in Line 3

Answer >>>

What is the output of the following code?

```
spam = {}
spam[1] = [1, 2]
spam[2] = [3, 4]
print(type(spam))
```

( ) <class 'list'>
( ) <class 'tuple'>
( ) <class 'set'>
( ) <class 'dict'>

Answer >>>

What is the output of the following code?

```
a, b = 10, 20
```

```
print({True: a, False: b} [a < b])
```

( ) 10
( ) 20
( ) KeyError: True
( ) Invalid Syntax

Answer >>>

What is the output of the following code?

```
spam = { 'z':'x', 'x':'y', 'y':'z' }
ham = 'x'
for x in range(len(spam)):
    ham = spam[ham]
    print(ham, end="")
```

( ) xyz
( ) yz
( ) yzx
( ) zxy
( ) KeyError

Answer >>>

Select all options which will return False given the code below?

```
tel = {'rick': 123, 'morty': 456}
```

[ ] 'spam' in tel
[ ] tel['spam'] is not None
[ ] tel.get('spam') is not None
[ ] tel.get('spam', True) is not None
[ ] tel.key('spam') is not None

Answer >>>

What is the output of the following code?

```
spam = {}
spam['f1'] = {'b1':11, 'b2':12}
spam['f2'] = {'b1':21, 'b2':22}
for ham in spam.keys():
```

```
    print(ham, end=' ')
```

( ) f1 f2
( ) b1 b2
( ) f1 b1 b2 f2 b1 b2
( ) f1 f2 b1 b2
( ) TypeError: keys() takes exactly 1 argument (0 given)

Answer >>>

Select all options that will print the key and value pair of the spam of the following code?

```
spam = {'b1':11, 'b2':12}
```

[ ] for x in spam.items():print(x[0], x[1])
[ ] for x, y in spam.items():print(x, y)
[ ] for x in spam.values():print(x[0], x[1])
[ ] for x, y in spam.values():print(x, y)
[ ] for x in spam:print(x[0], x[1])

Answer >>>

What is the output of the following code?

```
>>> {False: 'No', 0: 'Nay', 0.0: 'Nope'}
```

( ) {False: 'No', 0: 'Nay', 0.0: 'Nope'}
( ) {False: 'No'}
( ) {False: 'Nope'}
( ) {False: 'No', 0: 'Nope'}
( ) {False: 'No', 0: 'Nay'}
( ) {0.0: 'No'}
( ) {0.0: 'Nope'}

Answer >>>

How will you call the functions **spam(), ham(), and eggs()** so you will output **spam ham eggs**?

```
def spam():
    print("spam", end=" ")
    def ham():
        print("ham", end=" ")
```

```
        def eggs():
            print("eggs")
            return None
        return eggs
    return ham
```

[ ] spam(); ham(); eggs()
[ ] spam()()()
[ ] x = spam(); y = x(); z = y()
[ ] spam().ham().eggs()
[ ] Invalid Syntax

Answer >>>

How many 'b's will be printed based on the output of the following code?

```
def spam(n):
    result = 'b'
    for i in range(n):
        result += result
        yield result
for s in spam(2):
    print(s, end="")
```

( ) SyntaxError: invalid syntax
( ) 0
( ) 2
( ) 4
( ) 6

Answer >>>

What is the output of the following code?

```
def foo():
    for x in range(5):
        yield x*x
for x in foo():
    print(x, end=" ")
```

( ) 0
( ) 0 1 2 3 4
( ) 0 1 4 9 16
( ) TypeError: 'int' object is not iterable

Answer >>>

What is the output of the following code?

```
def spam(d, k, v):
    d[k]=v
    print(d, end=' ')
print(spam({}, '0', 'value'))
```

( ) SyntaxError: invalid syntax
( ) {}
( ) {'0': 'value'}
( ) {'0': 'value'} None
( ) SyntaxError: dynamic constant assignment error

Answer >>>

What is the output of the following code?

```
def spam(x, sum):
    return sum if x == 0 else sum + spam(x-1, sum)
print(spam(3, 0))
```

( ) SyntaxError: invalid syntax
( ) 0
( ) 6
( ) RecursionError: maximum recursion depth exceeded in comparison

Answer >>>

What do you call **end** in the print function call below?

```
print(x, end='')
```

( ) named argument
( ) positional argument
( ) keyword argument
( ) arbitrary argument

Answer >>>

How will you call the code below if you want to print 1, 2, 3?

```
def spam(a, b, c=3):
    print(a, b, c)
```

[ ] spam(1,2)
[ ] spam(1,2,3)
[ ] spam(b=2,a=1)
[ ] spam(a=1,2,c=3)
[ ] spam(a=1,2,3)

Answer >>>

Which of the option(s) is valid based on the following code?

```
1 def spam(a, b, c=3):
2     print(a, b, c)
3 spam(1, 2, c=3,)
```

[ ] a, b are positional arguments
[ ] 1, 2 are positional arguments
[ ] 1 2 3 will be printed
[ ] 3 is a positional argument
[ ] Invalid Syntax on Line 3

Answer >>>

What is the output of the following code?

```
123
456
789
```

( ) No output
( ) 123456789
( ) SyntaxError: invalid syntax
( ) NameError: name 123 is not defined

Answer >>>

What is the output of the following code?

```
>>> 2 ** 3 ** 2
```

( ) SyntaxError: invalid syntax
( ) 12
( ) 64
( ) 128
( ) 256
( ) 512

Answer >>>

What is the output of the following code?

```
>>> 5 ** 0 ** 0
```

( ) SyntaxError: invalid syntax
( ) 1
( ) 5
( ) 0

Answer >>>

What is the output of the following code?

```
i = 10
while len(str(i)) > 5:
    i-=1
    print(i, end='')
else:
    i+=1
    print(i, end='')
```

( ) 98765
( ) 987656
( ) 11
( ) 11 .. 99999 will be printed

Answer >>>

What is the output of the following code?

```
>>> i = 0
>>> while i != 0: i -= 1
... else: i += 1
```

```
>>> i
```

( ) SyntaxError: invalid syntax
( ) 0
( ) 1
( ) 2

Answer >>>

What is the output of the following code?

```
i = 30
while i > 0 :
    i -= 10
    print('loop', end=' ')
    if i <= 10:
        print('break', end=' ')
        break
else:
    print("else", end=' ')
```

( ) loop loop break
( ) loop loop loop break
( ) loop loop break else
( ) loop loop loop break else

Answer >>>

Which option(s) below will output

```
Hello World!
```

given the following code?

```
<<< INSERT CODE HERE >>>
print('World')
```

( ) print('Hello')
( ) print('Hello', ' ')
( ) print('Hello', sep=' ')
( ) print('Hello', end=' ')

Answer >>>

Which option(s) will result in the output

```
0 1 2
```

Given the following code

```
c, b, a = 2, 1, 0
a, c = c, b
b = b - c
<<< INSERT CODE HERE >>>
print(a, b, c)
```

[ ] a, b, c = b, c, a
[ ] c, b, a = a, c, b
[ ] b, c, a = a, b, c
[ ] a, c, b = c, b, a

Answer >>>

What is the output of the following code?

```
>>> def upcase(text): return text.upper()
>>> x = upcase
>>> f = [str.lower, x, str.capitalize]
>>> def dofunc(f):
...     message = f('Hello')
...     print(message)
>>> dofunc(x)
```

( ) SyntaxError: invalid syntax
( ) Hello
( ) hello
( ) HELLO
( ) AttributeError: type object 'str' has no attribute 'capitalize'

Answer >>>

Which option prints **HELLO WORLD** given the following code?

```
>>> def dofunc(text, b):
...     def lowcase(): return text.lower()
...     def upcase(): return text.upper()
```

```
...     return upcase if b else lowcase
```

[ ] >>> dofunc('Hello World', True)
[ ] >>> dofunc('Hello World', True)()
[ ] >>> dofunc('Hello World', (True))()
[ ] >>> dofunc('Hello World', (False,))()

Answer >>>

Exam block #1: Modules and Packages (12%)

Objectives covered by the block (6 items)

* import variants; advanced qualifying for nested modules

What is the output of the following code if **spam.py** is run?

 # spam.py

```
print("spam", end=' ')
import ham
```

 # ham.py

```
import eggs
print("ham", end=' ')
```

 # eggs.py

```
print("eggs", end=' ')
```

( ) Syntax Error
(X) spam eggs ham
( ) spam ham
( ) eggs ham spam
( ) spam ham eggs

Explanation:

 # spam.py
print("spam", end=' ') #1 print **spam**
import ham #2 go to ham.py
 # ham.py
import eggs #3 go to eggs.py
print("ham", end=' ') #5 print **ham**
 # eggs.py
print("eggs", end=' ') #4 print **eggs**

Question >>>

How do you call the function ham() saved as **spam.py** below?

```
def ham():
    print("Hello World")
```

[ ] import spam; ham()
[ ] import spam.ham; ham()
[X] import spam; spam.ham()
[X] from spam import ham; ham()
[ ] import ham from spam; ham()

Explanation:
https://docs.python.org/3/tutorial/modules.html
Question >>>

***** import variants; advanced qualifying for nested modules

Given the following package layout

```
package/
    subpackage1/
        __init__.py
        moduleX.py
        moduleY.py
    subpackage2/
        moduleZ.py
    moduleA.py
```

Select all option(s) containing valid relative imports called from __init__.py
[X] from .moduleY import spam
[X] from .moduleY import spam as ham
[X] from ..subpackage1 import moduleY
[X] from ..subpackage2.moduleZ import eggs
[X] from ..moduleA import foo

Explanation:
https://docs.python.org/3/reference/import.html#package-relative-imports
Question >>>

How will you shorten the function call to spam() defined inside packageA.subpackageB.subpackageC.moduleD?
[ ] import packageA.subpackageB.subpackageC.moduleD
[X] import packageA.subpackageB.subpackageC.moduleD as p

[ ] import packageA.subpackageB.subpackageC.moduleD alias p
[X] from packageA.subpackageB.subpackageC.moduleD import *
[X] from packageA.subpackageB.subpackageC.moduleD import spam
[X] from packageA.subpackageB.subpackageC.moduleD import spam as s
[ ] from packageA.subpackageB.subpackageC.moduleD import spam alias s

Explanation:
https://docs.python.org/3/tutorial/modules.html#more-on-modules
import packageA.subpackageB.subpackageC.moduleD is valid but it will not shorten the function call. alias is not part of the syntax for import.

Question >>>

***** dir(); sys.path variable

Select all valid parameters to function dir()
[X] No parameter
[X] Object
[X] 0
[X] None

Explanation:
https://docs.python.org/3/library/functions.html#dir

Question >>>

Select all valid option(s) about the result of dir()
[ ] A list of filenames inside the directory
[X] A list of the module's attribute
[X] A list of names of class attributes
[X] A list of names of object attributes
[X] A list of names of the base class attributes

Explanation:
https://docs.python.org/3/library/functions.html?#dir

Question >>>

***** dir(); sys.path variable

Select all valid option(s) about sys.path
[ ] sys.path is a string that specifies the path where Python is installed
[ ] sys.path is a string that specifies the path of the compiled Python bytecode
[X] sys.path is a list of strings that specifies the search path for modules

[X] A program is free to modify sys.path for its own purpose.
Explanation:
https://docs.python.org/3/library/sys.html#sys.path
Question >>>

***** math: ceil(), floor(), trunc(), factorial(), hypot(), sqrt(); random: random(), seed(), choice(), sample()

What is the output of the following code?

```
>>> math.ceil(-1.1)
```

(X) -1
( ) -1.0
( ) -2
( ) -2.0
Explanation:
https://docs.python.org/3/library/math.html#math.ceil
e.g. math.ceil(-1.1) is -1 because -1 > -1.1 and not -2 because -2 < -1.1.
Question >>>

***** math: ceil(), floor(), trunc(), factorial(), hypot(), sqrt(); random: random(), seed(), choice(), sample()

What is the output of the following code?

```
>>> math.floor(-1.1)
```

( ) -1
( ) -1.0
(X) -2
( ) -2.0
Explanation:
https://docs.python.org/3/library/math.html#math.floor
e.g. math.floor(-1.1) is -2 because -2 < -1.1 and not -1 because -1 > -1.1
Question >>>

***** math: ceil(), floor(), trunc(), factorial(), hypot(), sqrt(); random: random(), seed(), choice(), sample()

What is the output of the following code?

```
>>> math.factorial(3.0))
```

(X) 6
( ) 6.0
( ) TypeError: type float doesn't define __factorial__ method
( ) TypeError: factorial() takes 2 arguments

Explanation:
https://docs.python.org/3/library/math.html#math.factorial

Question >>>

What is the output of the following code?

```
>>> math.factorial(-3.0)
```

( ) -6
( ) -6.0
( ) TypeError: type float doesn't define __factorial__ method
(X) ValueError: factorial() not defined for negative values

Explanation:
https://docs.python.org/3/library/math.html#math.factorial

Question >>>

***** math: ceil(), floor(), trunc(), factorial(), hypot(), sqrt(); random: random(), seed(), choice(), sample()

What is the output of the following code?

```
>>> math.hypot(2)
```

( ) 3.6055512754639896
(X) 2.0
( ) TypeError: type int doesn't define __hypot__ method
( ) TypeError: hypot() takes 2 arguments

Explanation:
https://docs.python.org/3/library/math.html#math.hypot
e.g. math.sqrt(sum([2**2])) == 2.0

Question >>>

***** math: ceil(), floor(), trunc(), factorial(), hypot(), sqrt(); random: random(), seed(), choice(), sample()

What is the output of the following code?

```
>>> math.sqrt(1)
```

( ) 0.5
( ) 1
(X) 1.0
( ) TypeError: type int doesn't define __sqrt__ method

Explanation:
https://docs.python.org/3/library/math.html#math.sqrt
>>> import math
>>> type(math.sqrt(1))
<class 'float'>

Question >>>

***** math: ceil(), floor(), trunc(), factorial(), hypot(), sqrt(); random: random(), seed(), choice(), sample()

Select all option(s) which returns a random floating number between 0 and 1?
( ) math.random()
( ) math.random(1.0)
(X) random.random()
( ) random.random(1.0)

Explanation:
https://docs.python.org/3/library/random.html#random.random

Question >>>

Select all option(s) which returns a random number between 0 and 100?
( ) random.random(100)
( ) random.random(0, 100)
(X) random.random()*100
( ) random.random(100.0)

Explanation:
https://docs.python.org/3/library/random.html#random.random
e.g.
random.random()*(100-0)+0 == random number between 0 and 100
random.random()*(95-5)+5 == random number between 5 and 95

Question >>>

***** math: ceil(), floor(), trunc(), factorial(), hypot(), sqrt(); random:

random(), seed(), choice(), sample()

What can be the possible output of the following code?

```
random.seed(10, 2)
print(random.random())
```

( ) 3.6055512754639896
(X) 0.5714025946899135
( ) AttributeError: module 'random' has no attribute 'seed'
( ) TypeError: seed() takes 1 argument

Explanation:
https://docs.python.org/3/library/random.html#random.seed

Question >>>

***** math: ceil(), floor(), trunc(), factorial(), hypot(), sqrt(); random: random(), seed(), choice(), sample()

Select all option(s) to properly call the choice() and/or choices() function?

[ ] random.choice("spam", "ham", "eggs")
[X] random.choice(["spam", "ham", "eggs"])
[ ] random.choice({"spam", "ham", "eggs"})
[X] random.choices(["spam", "ham", "eggs"])
[X] random.choices(["spam", "ham", "eggs"], weights = [10, 1, 1], k = 14)

Explanation:
https://docs.python.org/3/library/random.html#random.choice

Question >>>

***** math: ceil(), floor(), trunc(), factorial(), hypot(), sqrt(); random: random(), seed(), choice(), sample()

What can be the possible output of the following code?

```
>>> random.sample(["spam", "ham", "eggs"], k = 1)
```

( ) spam
(X) [spam]
( ) TypeError: sample() got an unexpected keyword argument 'k'
( ) TypeError: sample() takes 1 argument

Explanation:

https://docs.python.org/3/library/random.html#random.sample
>>> import random
>>> type(random.sample(["spam", "ham", "eggs"], k = 1))
<class 'list'>
Question >>>

***** platform: platform(), machine(), processor(), system(), version(), python_implementation(), python_version_tuple()

Select all option(s) to properly call the platform() function?
[ ] system.platform()
[X] platform.platform()
[ ] system.platform(aliased=0, terse=0)
[ ] platform.platform(alias=0, version=0)
[X] platform.platform(aliased=0, terse=0)
Explanation:
https://docs.python.org/3/library/platform.html#platform.platform
Question >>>

***** platform: platform(), machine(), processor(), system(), version(), python_implementation(), python_version_tuple()

Select all option(s) to properly call the machine() function?
[ ] system. machine()
[X] platform. machine()
[ ] system. machine(aliased=0)
[ ] platform. machine (terse=0)
[ ] platform. machine (None)
Explanation:
https://docs.python.org/3/library/platform.html#platform.machine
Question >>>

***** platform: platform(), machine(), processor(), system(), version(), python_implementation(), python_version_tuple()

Select all option(s) to properly call the processor() function?
[ ] system.processor()
[X] platform.processor()

[ ] system.processor(aliased=0)
[ ] platform.processor(terse=0)
[ ] platform.platform(None)
Explanation:
https://docs.python.org/3/library/platform.html#platform.processor
Question >>>

***** platform: platform(), machine(), processor(), system(), version(), python_implementation(), python_version_tuple()

Select all option(s) to properly call the system() function?
[ ] system.system()
[X] platform.system()
[ ] system.system(aliased=0)
[ ] platform.system(terse=0)
[ ] platform.system(None)
Explanation:
https://docs.python.org/3/library/platform.html#platform.system
Question >>>

Select all valid option(s) about system() function
[X] system() returns the OS hosting Python
[ ] system() returns the execution environment of Python
[X] Possible return values are **Linux**, **Darwin**, **Java**, **Windows** or an empty string if it can't be determined.
[ ] Possible return values are **CPython**, **IronPython**, **Jython**, **PyPy**.
Explanation:
https://docs.python.org/3/library/platform.html#platform.system
Question >>>

***** platform: platform(), machine(), processor(), system(), version(), python_implementation(), python_version_tuple()

Select all option(s) to properly call the version() function?
[ ] system.version()
[X] platform.version()
[ ] system.version(aliased=0)
[ ] platform.version(terse=0)

[ ] platform.version(None)

Explanation:

https://docs.python.org/3/library/platform.html#platform.version

Question >>>

What is the datatype of the return value of the function platform.version()?

( ) int

( ) float

(X) str

( ) array

Explanation:

https://docs.python.org/3/library/platform.html#platform.version

```
>>> from platform import version
>>> type(version())
<class 'str'>
```

Question >>>

***** platform: platform(), machine(), processor(), system(), version(), python_implementation(), python_version_tuple()

Select all option(s) to properly call the python_implementation() function?

[X] system.python_implementation()

[ ] platform.python_implementation()

[ ] system.python_implementation(aliased=0)

[ ] platform.python_implementation(terse=0)

[ ] platform.python_implementation(None)

Explanation:

https://docs.python.org/3/library/platform.html#platform.python_implementat

Question >>>

Select all option(s) about the python_implementation() that is TRUE?

[ ] python_implementation() returns the OS hosting Python

[X] python_implementation() returns the execution environment of Python

[ ] Possible return values are **Linux**, **Darwin**, **Java**, **Windows** or an empty string if it can't be determined.

[X] Possible return values are **CPython**, **IronPython**, **Jython**, **PyPy**.

Explanation:

https://docs.python.org/3/library/platform.html#platform.python_implementat
Question >>>

***** platform: platform(), machine(), processor(), system(), version(), python_implementation(), python_version_tuple()

Select all option(s) to properly call the python_version_tuple() function?

[ ] system.python_version_tuple()
[X] platform.python_version_tuple()
[ ] system.python_version_tuple(aliased=0)
[ ] platform.python_version_tuple(terse=0)
[ ] platform.python_version_tuple(None)

Explanation:
https://docs.python.org/3/library/platform.html#platform.python_version_tupl
Question >>>

***** idea: __pycache__, __name__, public variables, __init__.py

Which of the statements below is valid?

[ ] Python is interpreted therefore it never compiles the **py** files.
[X] Python is interpreted however it compiles the **py** file into **pyc** file.
[ ] Compiled Python files is stored inside the **__pyc__** folder
[X] Compiled Python files is stored inside the **__pycache__** folder
[ ] Compiled Python files is stored inside the **__cache__** folder

Explanation:
Python caches the compiled version of each module in the **__pycache__** directory under the name **module.*version*.pyc**.
Question >>>

The extension of a compiled bytecode of the Python source file is

( ) .py
(X) .pyc
( ) .__pycache__
( ) Python is an interpreted language hence it does not compile the source file

Explanation:
Python caches the compiled version of each module in the **__pycache__** directory under the name **module.*version*.pyc**.

Question >>>

***** idea: __pycache__, __name__, public variables, __init__.py

Select all valid option(s) about __name__
[ ] The __name__ is a built-in constant and can't be modified
[X] The __name__ is a built-in variable and can be modified
[ ] The __name__ by default is None and must be set
[X] If the source is the main program, the interpreter sets __name__ to "__main__"
[X] If the file is imported from another module, __name__ will be set with the module's name.
Explanation:
https://docs.python.org/3/reference/import.html#__name__

Question >>>

***** idea: __pycache__, __name__, public variables, __init__.py

How should you write the variable spam to inform a module user that it should not be accessed directly?
[ ] spam since all variables in modules are considered private
[X] _spam
[X] __spam
[ ] SPAM
Explanation:
https://docs.python.org/3/tutorial/classes.html#private-variables

Question >>>

***** idea: __pycache__, __name__, public variables, __init__.py

Select all valid option(s) about __init__.py
[X] __init__.py is contained in regular packages
[ ] __init__.py is contained in namespace packages
[X] __init__.py is automatically executed when the regular package is imported.
[ ] __init__.py is automatically executed when the namespace package is imported
Explanation:
https://docs.python.org/3/reference/import.html#regular-packages

Question >>>

***** searching for modules/packages; nested packages vs directory tree

What directories are searched by the interpreter for spam.py given the code below?

```
import spam
print(spam.ham)
print(spam.eggs)
```

[X] Directory where spam.py was run
[X] Current directory if the interpreter is run interactively
[ ] List of directories contained in PATH environment variable
[X] List of directories contained in PYTHONPATH environment variable
[X] Python installation-dependent directories configured during installation
[X] List of directories in sys.path

Explanation:
When a module named **spam** is imported, the interpreter first searches for a built-in module with that name. If not found, it then searches for a file named **spam.py** in a list of directories given by the variable **sys.path**. **sys.path** is initialized from these locations:

- The directory containing the input script (or the current directory when no file is specified).
- **PYTHONPATH** (a list of directory names, with the same syntax as the shell variable PATH).
- The installation-dependent default.

Question >>>

Exam block #2: Exceptions (14%)

Objectives covered by the block (5 items)

* except, except:-except; except:-else:, except (e1,e2)

What is the output of the following code?

```
try:
    abcd
    efgh
except:
    pass
```

(X) No output
( ) SyntaxError: invalid syntax
( ) NameError: name 'UndefinedException' is not defined
( ) Add () on Line 2 and 3 to fix the syntax error

Explanation:

```
try:
        abcd  #1 raises NameError
except:
        efgh
except:     #2 execute next line
        pass  #3 do nothing
```

Question >>>

What is the output of the following code?

```
>>> try:
...     raise OSError
... finally:
...     pass
```

( ) No output
(X) OSError
( ) NameError: name 'OSError' is not defined
( ) SyntaxError: invalid syntax

Explanation:

https://docs.python.org/3/tutorial/errors.html?#defining-clean-up-actions

Question >>>

What is the output of the following code?

```
try:
    raise ValueError
except TypeError, ValueError:
    raise
```

( ) No output
( ) TypeError
( ) ValueError
(X) SyntaxError: invalid syntax

Explanation:
The correct code specifies a tuple of exceptions.
e.g.
...
except (TypeError, ValueError):
…

Question >>>

***** except, except:-except; except:-else:, except (e1,e2)

What will happen if spam.py is run?
 # spam.py

```
try:
    print(x)
except:
    print("An exception occurred")
```

( ) the script will run but will not print anything
( ) **None** will be printed
(X) **An exception occurred** will printed
( ) Compile time error

Explanation:
>>> try:
... print(x) #1 raise NameError: name 'x' is not defined go to except:
>>> except: #2 execute next line

... print("An exception occurred") #3 prints **An exception occurred**

Question >>>

What is the output of the following code?

```
>>> def this_fails():
...     x = 1/0
>>> try:
...     this_fails()
... except ZeroDivisionError:
...     pass
```

(X) No output
( ) SyntaxError: invalid syntax
( ) ZeroDivisionError: division by zero
( ) NameError: name 'ZeroDivisionError' is not defined

Explanation:
https://docs.python.org/3/tutorial/errors.html#handling-exceptions

Question >>>

***** except, except:-except; except:-else:, except (e1,e2)

Which option(s) will print **ELSE** given the following code?

```
try:
    <<< INSERT CODE HERE >>>
except ZeroDivisionError:
    print('ZeroDivisionError')
except TypeError:
    print('TypeError')
else:
    print('ELSE')
```

[ ] raise Exception
[ ] raise ZeroDivisionError
[ ] raise TypeError
[ ] raise
[X] pass
[ ] replace <<< INSERT CODE HERE >>> with blank

Explanation:
ELSE will only be printed if raise is not called. Replacing **<<< INSERT CODE HERE >>>** with blank results in error because the try: block is not optional.

Question >>>

What is the output of the following code?

```
try:
    print("1", end='')
    raise Exception
    print("2", end='')
except BaseException:
    print("3", end='')
else:
    print("4", end='')
finally:
    print("5")
```

( ) NameError: name 'BaseException' is not defined
( ) 1235
( ) 1245
(X) 135
( ) 145

Explanation:

```
try:
    print("1", end='')  #1 Print 1
    raise Exception     #2 raise Exception go to except BaseException:
    print("2", end='')
except BaseException:   #3 execute next
    print("3", end='')  #4 Print 3 go to finally:
else:
    print("4", end='')
finally:                #5 execute next
    print("5")          #6 Print 5
```

Question >>>

What is the output of the following code?

```
class E(Exception):
    def __init__(self, message):
        self.message = message
    def __str__(self):
        return "Surprise"


try:
    raise Exception("Stop")
except E as e:
    print(e)
else:
    print("Goodbye")
```

(X) Unhandled Exception
( ) Surprise
( ) Stop
( ) Goodbye

Explanation:
https://docs.python.org/3/tutorial/errors.html#handling-exceptions
Question >>>

***** except, except:-except; except:-else:, except (e1,e2)

What is the output of the following code?

```
try:
    raise Exception
except:
    print("Spam", end='')
except BaseException:
    print("Ham", end='')
except Exception:
    print("Eggs")
```

( ) Eggs
( ) Spam Eggs
( ) Spam Ham Eggs

(X) Syntax Error

Explanation:

SyntaxError because default 'except:' clause must be last defined.

Question >>>

If there are more than 1 except clause, what happens after a try clause executes?

[X] None of the except is executed

[ ] At least 1 except is executed

[X] Not more than 1 except is executed

[ ] Exactly 1 of the except is executed

Explanation:

https://docs.python.org/3/tutorial/errors.html?#handling-exceptions

Question >>>

***** the hierarchy of exceptions

Which of the statements below is valid?

[X] The finally branch in a try block is always executed.

[ ] The finally branch in a try block will only be executed if an exception occurs.

[ ] The finally branch in a try block will only be executed if the exception did not occur

[X] The finally branch in a try block is optional

[ ] The finally branch in a try block is required because it is always executed.

Explanation:

https://docs.python.org/3/reference/compound_stmts.html#finally

Question >>>

What is the output of the following code?

```
class Spam(Exception):
    pass
class Ham(Spam):
    pass
for cls in [Spam, Ham]:
    try:
        raise cls()
```

```
    except Spam:
        print("Spam", end=" ")
    except Ham:
        print("Ham", end=" ")
```

( ) Spam Ham
(X) Spam Spam
( ) Spam Ham Spam Ham
( ) Invalid Syntax

Explanation:
A class in the except clause is compatible with an exception if it is the same class or a base class thereof. The raised Spam and Ham exceptions will both go to except Spam: since both classes are compatible with Spam.
Question >>>

Which option are valid replacements for the marker in the given code?

```
>>> try:
...     x = 1/0
... <<< INSERT CODE HERE >>>
...     pass
```

[X] except BaseException:
[X] except Exception:
[ ] except MathError:
[ ] except ArithmeticException:
[X] except ArithmeticError:
[ ] except DivisionZeroError:
[X] except ZeroDivisionError:

Explanation:
https://docs.python.org/3/library/exceptions.html#exception-hierarchy
Question >>>

***** raise, raise ex, assert

What is the output of the following code if spam.txt does not exist?

```
import sys
try:
    f = open('spam.txt')
```

```
    s = f.readline()
except:
    raise
```

( ) the script will run but will not print anything
( ) "None" will be printed
(X) FileNotFoundError: [Errno 2] No such file or directory: 'spam.txt'
( ) Compile time error

Explanation:

```
import sys
try:
        f = open('spam.txt')  #1 raise FileNotFound go to except:
        s = f.readline()
except:                 #2 execute next
        raise            #3 raise FileNotFound
```

Question >>>

***** raise, raise ex, assert

Select which option will call the __init__ method of Exception based on the code below.

```
class SpamException(Exception):
    def __init__(self, message):
        <<< INSERT CODE HERE >>>
        self.message = message
raise SpamException("Spam")
```

[X] super().__init__(message)
[ ] Exception.__init__(self, message)
[X] super(SpamException, self).__init__(message)
[ ] super.__init__(message)

Explanation:

https://docs.python.org/3/library/functions.html#super

Question >>>

What is the output of the following code?

```
try:
    raise UndefinedException
```

```
except:
    pass
```

[X] No output
[ ] SyntaxError: invalid syntax
[ ] NameError: name 'UndefinedException' is not defined
[ ] Add () on Line 2 to fix the syntax error

Explanation:

try:
    raise UndefinedException #1 raise NameError: name 'UndefinedException' is not defined
except: #2 execute next
    pass   #3 do nothing

Question >>>

What is the output of the following code?

```
try:
    raise UndefinedException
except NameError:
    print('NameError')
except UndefinedException:
    print('UndefinedException')
except:
    pass
```

( ) No output
(X) NameError
( ) UndefineException
( ) SyntaxError: invalid syntax

Explanation:

try:
    raise UndefinedException #1 raise NameError: name 'UndefinedException' is not defined
except NameError: #2 execute next
    print('NameError')   #3 print **NameError**
except UndefinedException:
    print('UndefinedException')

except:
    pass

Question >>>

What is the output of the following code?

```
try:
    raise IOError
except IOError:
    raise RuntimeError from None
```

( ) No output
( ) IOError
(X) RuntimeError
( ) SyntaxError: invalid syntax

Explanation:

try:
    raise IOError #1 raise IOError
except IOError: #2 execute next
    raise RuntimeError from None #3 raise RuntimeError

Question >>>

What is the output of the following code?

```
try:
    raise IOError
except IOError as e:
    raise RuntimeError from e
```

( ) No output
( ) IOError
(X) RuntimeError
( ) SyntaxError: invalid syntax

Explanation:

try:
    raise IOError #1 raise IOError
except IOError as e: #2 execute next
    raise RuntimeError from e #3 raise RuntimeError

Question >>>

* raise, raise ex, assert

Which of the statements below is valid?

```
spam = 0
assert spam == 0
```

( ) AssertionError will be triggered because the expression is True
(X) No AssertionError will be triggered since the expression is True
( ) Missing parentheses in call to assert error will be displayed
( ) The word True will be printed on screen

Explanation:
AssertionError is raised when False not True

Question >>>

What is the result of the following code?

```
>>> assert(False, 'Trigger Assertion')
```

[X] No output
[ ] Trigger Assertion
[ ] SyntaxError: invalid syntax
[X] Assertion is always true

Explanation:
The assertion is always True because of the parentheses. Python treats (False, 'Trigger Assertion') as a non-empty tuple which it evaluates to True.

Question >>>

* event classes, except E as e, arg property
* event classes, except E as e, arg property

Which option(s) are valid except clause for ZeroDivisionError to be accessed as variable e?

[X] except ZeroDivisionError as e:
[ ] except ZeroDivisionError(e):
[X] except (ZeroDivisionError) as e:
[ ] except ZeroDivisionError e:
[ ] except (ZeroDivisionError as e):

Explanation:
https://docs.python.org/3/tutorial/errors.html#handling-exceptions

Question >>>

What is the output of the following code?

```
try:
    a = 1/'0'
except (ZeroDivisionError, TypeError) as e:
    print(type(e))
```

( ) The script will run but will not print anything
( ) <class 'ZeroDivisionError'>
(X) <class 'TypeError'>
( ) Invalid Syntax

Explanation:
'0' is a string not a number. 1/'0' results TypeError
Question >>>

***** event classes, except E as e, arg property

Which option will print ('spam', 'eggs') based on the following code?

```
try:
    raise Exception('spam', 'eggs')
except Exception as exception:
    <<< INSERT CODE HERE >>>
```

[ ] print(exception.params)
[X] print(exception)
[X] print(exception.args)
[ ] print(exception.iterable[:])

Explanation:
print(exception) # prints .args via __str__() e.g. **('spam', 'eggs')**
print(exception.args) # prints .args e.g. **('spam', 'eggs')**
Question >>>

What is the output of the following code?

```
>>> type(Exception().args)
```

( ) <class 'str'>
( ) <class 'list'>
(X) <class 'tuple'>
( ) <class 'dict'>

Explanation:
args is a tuple of arguments given to the exception constructor.
Question >>>

What is the output of the following code?

```
>>> try:
...     raise Exception('spam', 'eggs')
... except Exception as inst:
...     x, y = inst.args
>>> x, y
```

(X) ('spam', 'eggs')
( ) ValueError: too many values to unpack (expected 2)
( ) TypeError: 'tuple' object does not support item assignment
( ) SyntaxError: invalid syntax

Explanation:
https://docs.python.org/3/library/exceptions.html#BaseException.args
Question >>>

***** self-defined exceptions, defining and using

What is the output of the following code?

```
class AgeException(Exception):
    def __init__(self, age):
        super(AgeException, self).__init__("AgeException")
try:
    raise AgeException(16)
except AgeException as e:
    print(e)
```

( ) The script will run but will not print anything
(X) AgeException will be printed
( ) TypeError: super() argument 1 must be type
( ) TypeError:__init__() argument 1 must be type

Explanation:
>>> class AgeException(Exception):
... def __init__(self, age): #2 initialize ArgException

```
...         super(AgeException, self).__init__("AgeException")
>>> try:
...     raise AgeException(16)    #1 raise AgeException
>>> except AgeException as e:     #3 execute next
...     print(e)                  #4 print AgeException
```

Question >>>

***** self-defined exceptions, defining and using

What is the output of the following code?

```
class MyException(Exception):
    pass
try:
    raise MyException("spam", "ham", "eggs")
except MyException as s:
    print(s)
```

( ) The script will run but will not print anything
( ) spam ham eggs
(X) ('spam', 'ham', 'eggs')
( ) TypeError: expected Exception not type

Explanation:

print(s) # prints .args via s.__str__() e.g. **("spam", "ham", "eggs")**

Question >>>

Exam block #3: Strings (18%)

Objectives covered by the block (8 items)

* ASCII, UNICODE, UTF-8, codepoints, escape sequences

Select all valid option(s) below about string
[X] string.ascii_letters is a concatenation of ascii_lowercase and ascii_uppercase
[ ] string.ascii_letters is a concatenation of ascii_lowercase, ascii_uppercase and digits
[ ] string.ascii_letters are all printable characters found in the keyboard
[X] string.ascii_lowercase contains 'abcdefghijklmnopqrstuvwxyz'
[X] string.ascii_uppercase contains 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
Explanation:
https://docs.python.org/3/library/string.html#module-string
Question >>>

* ASCII, UNICODE, UTF-8, codepoints, escape sequences
* ASCII, UNICODE, UTF-8, codepoints, escape sequences
* ASCII, UNICODE, UTF-8, codepoints, escape sequences
* ASCII, UNICODE, UTF-8, codepoints, escape sequences

What is the output of the following code?

```
print("\\\\")
```

( ) Syntax Error
( ) \\\\
( ) \\\
(X) \\
( ) \
Explanation:
"\\" is 1 literal backslash and should come in pairs. e.g. "\\\\" are 2 literal backslashes
Question >>>

What is the output of the following code?

```
print("\\\\\")
```

(X) Syntax Error
( ) \\\\\
( ) \\\\
( ) \\\
( ) \\
( ) \

Explanation:
"\\" is 1 literal backslash and should come in pairs. e.g. "\\\\\" have an excess backslash which do not have a matching pair

Question >>>

What is the output of the following code?

```
print("C:\Program Files\Microsoft\Windows NT", end="")
print("\")
```

(X) Syntax Error
( ) C:\Program Files\Microsoft\Windows NT\
( ) Replace escaped characters with "?" e.g. C:?rogram Files?icrosoft? indows NT?
( ) Ignore escaped characters e.g. C: rogram Filesicrosoftindows NT

Explanation:
The character "\" inside a string is used to escape special characters. The literal backslash should be represented as "\\"

Question >>>

What is the output of the following code?

```
print("\\/\\//", len("\\/\\//"))
```

( ) \\/\\// 8
( ) \/\// 8
(X) \/\// 6
( ) //// 4
( ) Syntax Error

Explanation:
The literal backslash is represented as "\\" and will count as only 1 character.

Question >>>

What will be printed in the following code?

```
spam = """""""""
ham = """"
""""
print(spam, ham)
```

( ) Syntax Error
( ) Two empty strings
(X) An empty string and a new line character
( ) Two new line character

Explanation:
String literals can span multiple lines. One way is using triple-quotes: """..."""" or '''...'''. End of lines are automatically included in the string

Question >>>

***** ord(), chr(), literals

What is the output of the following code?

```
spam = chr('a')
ham = ord(spam)
print(spam, ham)
```

( ) 97 a
(X) TypeError: an integer is required (got type str)
( ) TypeError: chr() takes exactly two arguments (1 given)
( ) TypeError: ord() takes exactly two arguments (1 given)
( ) Syntax Error

Explanation:
chr(i) Return the string representing a character whose Unicode code point is the integer i.
e.g. chr(97) returns **'a'**; chr('a') results TypeError
ord(c) Given a string representing one Unicode character, return an integer representing the Unicode code point of that character.
e.g. ord('a') returns **97**; ord(97) results TypeError

Question >>>

***** ord(), chr(), literals

What is the output of the following code?

```
"spam"
"ham"
"eggs"
print("Hello World")
```

( ) spam ham eggs Hello World
(X) Hello World
( ) SyntaxError: invalid syntax
( ) NameError: name 'spam' is not defined

Explanation:
Literals without any associated actions are ignored.

Question >>>

Which option will print the following output?

```
John said: "I'm fine!"
```

[X] print('John said: "I\'m fine!"')
[X] print("John said: \"I'm fine!\"")
[ ] print("John said: ""I'm fine!""")
[ ] print('John said: "I"m fine!"')
[X] print('John said: \"I\'m fine!\"')

Explanation:
https://docs.python.org/3/tutorial/introduction.html#strings
>>> print('John said: "I\'m fine!"')
John said: "I'm fine!"
>>> print("John said: \"I'm fine!\"")
John said: "I'm fine!"
>>> print("John said: ""I'm fine!""") # double-quote not printed
John said: I'm fine!
>>> print('John said: "I"m fine!"')   # single-quote not printed
John said: "Im fine!"
>>> print('John said: \"I\'m fine!\"')
John said: "I'm fine!"

Question >>>

***** indexing, slicing, immutability

Which option will return a different result given the code below?

```
s = 'Python'
```
( ) print(s[0] + s[-1])
( ) print(s[::5])
(X) print(s[::-5])
( ) print(s[::-1][::-5])

Explanation:
All options will result in **Pn** except for print(s[::-5]) which results in **nP.**
TIP: Once you have a different result no need to check the remaining option(s)
Question >>>

***** indexing, slicing, immutability

Which option will return True given the following code?

```
spam = 'FuBar'
ham = spam[:]
```
[X] spam == ham
[X] id(spam) == id(ham)
[X] spam.startswith(ham)
[X] spam.endswith(ham)
[ ] spam.equals(ham)

Explanation:
spam == ham **# True**
id(spam) == id(ham) **# True**
spam.startswith(ham) **# True**
spam.endswith(ham) **# True**
spam.equals(ham) **# AttributeError: 'str' object has no attribute 'equals'**
Question >>>

Which option(s) will return **Ham**
[X] 'Spam,Ham,Eggs'[5:8]
[X] 'Spam,Ham,Eggs'[-8:-5]
[X] 'Spam,Ham,Eggs'[5:-5]
[ ] 'Spam,Ham,Eggs'[-5:-8]
[ ] 'Spam,Ham,Eggs'[-5:5]

Explanation:

'Spam,Ham,Eggs'[5:8]   **# Ham**
'Spam,Ham,Eggs'[-8:-5] **# Ham**
'Spam,Ham,Eggs'[5:-5] **# Ham**
'Spam,Ham,Eggs'[-5:-8] **# Empty string**
'Spam,Ham,Eggs'[-5:5] **# Empty string**

Question >>>

***** indexing, slicing, immutability

What is the output of the following code?

```
spam = 'spam'
print(spam[0], end=' ')
spam[0]='x'
print(spam)
```

( ) No output
( ) s xpam
( ) s spam
(X) s followed by TypeError: 'str' object does not support item assignment

Explanation:
Python strings cannot be changed — they are immutable. Therefore, assigning to an indexed position in the string results in an error.

Question >>>

What is the output of the following code?

```
>>> s = 'Hello World'
>>> for i in len(s):
...     s[i] = s[i].upper()
>>> s
```

( ) Hello World
( ) HELLO WORLD
( ) TypeError: 'str' object does not support item assignment
(X) TypeError: 'int' object is not iterable

Explanation:
https://docs.python.org/3/tutorial/controlflow.html?#for-statements

Question >>>

***** iterating through,

What is the output of the following code?

```
s = '0123456789'
print(s[::2], s[:-2:2], s[2::2])
```

( ) 01 89 23
( ) 01 01 23
(X) 02468 0246 2468
( ) 02468 8 0
( ) SyntaxError: invalid syntax

Explanation:
default start=**0**; default end=**10** (length of the string); default step=**1**
s[::2]   # start=**0**, end=**10**,   step=**2** results **02468**
s[:-2:2] # start=**0**, end=**10-2**=**8**, step=**2** results **0246**
s[2::2] # start=**2**, end=**10**,   step=**2** results **2468**

Question >>>

***** concatenating, multiplying, comparing (against strings and numbers)

What is the output of the following code if the user enters 1 on the first prompt and 2 on the second prompt?

```
a = input("Enter first number:")
b = input("Enter second number:")
print(a + b)
```

( ) TypeError: input() takes 0 positional arguments but 1 was given
( ) 3
(X) 12
( ) TypeError: unsupported operand type(s) for +: 'str' and 'str'

Explanation:
'1' + '2' == '12' or the concatenation of the strings '1' and '2'

Question >>>

What is the output of the following code?

```
foo = [
    'Spam',
```

```
    'Ham'
    'Eggs'
]
print(foo)
```

( ) ['Spam', 'Ham', 'Eggs']
(X) ['Spam', 'HamEggs']
( ) ['Spam']
( ) SyntaxError: invalid syntax

Explanation:
https://docs.python.org/3/reference/lexical_analysis.html#string-literal-concatenation

Question >>>

***** concatenating, multiplying, comparing (against strings and numbers)

What is the output of the following code?

```
>>> None * 2
```

( ) 0
( ) None
( ) NoneNone
(X) TypeError: unsupported operand type(s) for *

Explanation:
The None object has no arithmetic operators defined.

Question >>>

What is the output of the following code?

```
>>> spam, ham = 1, "ham"
>>> spam *= 3
>>> ham *= 3
>>> spam, ham
```

( ) SyntaxError: invalid syntax
(X) (3, hamhamham)
( ) (3, 0)
( ) TypeError: unsupported operand type(s) for *=: 'str' and 'int'

Explanation:
s * n or n * s is equivalent to adding s to itself n times
>>> "ham" * 3
'**hamhamham**'
Question >>>

What is the output of the following code?

```
>>> 2 * 'DUN-' + 'DUUUUN!!!'
```

( ) SyntaxError: invalid syntax
(X) DUN-DUN-DUUUUN!!!
( ) 2
( ) TypeError: unsupported operand type(s) for * 'int' and 'str'

Explanation:
s * n or n * s is equivalent to adding s to itself n times.
>>> 2 * 'DUN-'
'**DUN-DUN-**'
>>> 'DUN-DUN-' + 'DUUUUN!!!'
'**DUN-DUN-DUUUUN!!!**'
Question >>>

What is the output of the following code?

```
>>> 2 * ('Yes' + 3*'!')
```

( ) 0
( ) 8
( ) SyntaxError: invalid syntax
( ) TypeError: unsupported operand type(s) for *: 'int' and 'str'
(X) 'Yes!!!Yes!!!'

Explanation:
>>> 3*'!'
'**!!!**'
>>> 'Yes' + '!!!'
'**Yes!!!**'
>>> 2 * 'Yes!!!'
'**Yes!!!Yes!!!**'
Question >>>

***** concatenating, multiplying, comparing (against strings and

numbers)

What is the output of the following code?

```
>>> sorted([5, "1", 100, "34"])
```

( ) ["1", 5, "34", 100]
( ) [5, "1", "34", 100]
( ) ["1", "100", "34", "5"]
( ) [1, 5, 34, 100]
(X) TypeError: '<' not supported between instances of 'str' and 'int'

Explanation:
There are data types that can't be compared to each other using just sorted() because they are too different. Python will return TypeError if you attempt to use sorted() on a list containing non-comparable data.

Question >>>

What is the output of the following code?

```
x = "0"
y = "1"
z = "2"
x = y < z
print(x == 1, type(x))
```

( ) False <class 'bool'>
(X) True <class 'bool'>
( ) False <class 'str'>
( ) True <class 'str'>

Explanation:
y < z results in <class 'bool'> overrides content of the variable x

Question >>>

***** in, not in

What is the output of the following code?
spam.txt

```
spam ham eggs
```

spam.py

```
f = open('spam.txt', 'r')
```

```
if 'eggs' in f:
    print('Eggs found')
else:
    print('Eggs not found')
```

( ) Eggs found
(X) Eggs not found
( ) TypeError: argument type TextIOWrapper not iterable
( ) SyntaxError: invalid syntax

Explanation:
open('spam.txt', 'r') opens the file and returns the **file object,** not the **file content**.

Question >>>

***** in, not in

Which of the option(s) below are valid given the following code?

```
>>> '' not in 'spam'
```

[ ] Prints True
[ ] Empty string is not in the string 'spam'
[X] Prints False
[X] Empty string is always part of any string no exception

Explanation:
'' in s is true for every string s: n is zero, so we are checking s[i:i]; and that is the empty string itself for every valid index i:

```
>>> s = 'spam'
>>> s[0:0]
''
>>> s[1:1]
''
>>> s[2:2]
''
>>> s[3:3]
''
```

Question >>>

***** .isxxx(), .join(), .split()

Which of the calls below are valid String function calls and will return True?
[X] 'abc123'.isalnum()
[X] 'abc'.isalpha()
[ ] '123abc'.isidentifier()
[X] '123abc'.islower()
[X] '123'.isdigit()
[X] 'Abc'.istitle()
Explanation:
https://docs.python.org/3/library/stdtypes.html?#string-methods
>>> 'abc123'.isalnum()
True
>>> 'abc'.isalpha()
True
>>> '123abc'.isidentifier()
False
>>> '123abc'.islower()
True
>>> '123'.isdigit()
True
>>> 'Abc'.istitle()
True
Question >>>

***** .isxxx(), .join(), .split()

What is the output of the following code?

```
>>> "/".join({"Month": "12", "Day": "25", "Year":"2021"})
```

( ) 12/25/2021
(X) Month/Day/Year
( ) Month/12/Day/25/Year/2021
( ) TypeError: can only join an iterable
Explanation:
https://docs.python.org/3/library/stdtypes.html#str.join
Iterating a dictionary calls __iter__() which iterates over the **keys** of a dictionary.
Question >>>

What is the output of the following code?

```
>>> "XYZ".join("123")
```

( ) XYZ123
( ) 123XYZ
(X) 1XYZ2XYZ3
( ) X123Y123Z
( ) TypeError: can only join an iterable

Explanation:
https://docs.python.org/3/library/stdtypes.html#str.join
Iterating a string calls __iter__() which iterates over each character of the string.

Question >>>

***** .isxxx(), .join(), .split()

What is the output of the following code?

```
>>> "/spam/ham/eggs/".split("/")
```

( ) ['spam', 'ham', 'eggs']
(X) ['', 'spam', 'ham', 'eggs', '']
( ) ('spam', 'ham', 'eggs')
( ) ('', 'spam', 'ham', 'eggs', '')

Explanation:
https://docs.python.org/3/library/stdtypes.html#str.split
>>> "spam/ham/eggs".split("/")
['spam', 'ham', 'eggs']
>>> "/spam".split("/")
['', 'spam']
>>> "eggs/".split("/")
['eggs', '']

Question >>>

***** .sort(), sorted(), .index(), .find(), .rfind()

What is the output of the following code?

```
>>> spam = [4*(3+5), 4*3+5, 4+3*5, (4+3)*5]
>>> spam.sort(reverse=True)
>>> spam
```

( ) TypeError: 'reverse' is an invalid keyword argument for sort()

( ) [32, 17, 19, 35]
( ) [17, 19, 32, 35]
(X) [35, 32, 19, 17]
( ) [35, 19, 17, 32]

Explanation:

>>> [4*(3+5), 4*3+5, 4+3*5, (4+3)*5]
[32, 17, 19, 35]
>>> _.sort(reverse=True)
>>> _
[35, 32, 19, 17]

Question >>>

***** .sort(), sorted(), .index(), .find(), .rfind()

What is the output of the following code?

```
d = { 'zero':0, 'one':1, 'three':3, 'two':2 }
for k in sorted(d.keys()):
    print(d[k], end=' ')
```

( ) TypeError: sorted expected 2 arguments, got 1
( ) 0 1 2 3
(X) 1 3 2 0
( ) zero one two three
( ) one three two zero

Explanation:

sorted(d.keys()) will sort the keys alphabetically or **"one"**, **"three"**, **"two"**, **"zero"**

Question >>>

What is the output of the following code?

```
>>> sorted(['banana', 'pear', 'grapes', 'apple'], key=lambda x: x[::-1])
```

( ) ['apple', 'banana', 'grapes', 'pear']
(X) ['banana', 'apple', 'pear', 'grapes']
( ) SyntaxError: invalid syntax
( ) TypeError: 'key' is an invalid keyword argument for sorted()

Explanation:

key=lambda x: x[::-1] reverses of the text or **'ananab', 'elppa', 'raep', 'separg'** to be used as keys for sorting

Question >>>

What is the output of the following code?

```
def reverse(word):
    return word[::-1]
print(sorted(['banana', 'pear', 'grapes', 'apple'], key=reverse))
```

( ) ['pear', 'grapes', 'banana', 'apple']
(X) ['banana', 'apple', 'pear', 'grapes']
( ) ['grapes', 'pear', 'apple', 'banana']
( ) TypeError: 'key' is an invalid keyword argument for sorted()

Explanation:
word[::-1] reverses the text or **'ananab', 'elppa', 'raep', 'separg'** to be used as keys for sorting

Question >>>

What is the output of the following code?

```
def reverse(word):
    return word[::-1]
print(sorted(['banana', 'pear', 'grapes', 'apple'],
key=reverse, reverse=True))
```

( ) ['apple', 'banana', 'grapes', 'pear']
(X) ['grapes', 'pear', 'apple', 'banana']
( ) ['banana', 'apple', 'pear', 'grapes']
( ) TypeError: 'key' is an invalid keyword argument for sorted()

Explanation:
word[::-1] reverses the text or **'ananab', 'elppa', 'raep', 'separg'** to be used as keys for sorting
reverse=True reverses the order to descending or **'separg', 'raep', 'elppa', 'ananab'**

Question >>>

***** .sort(), sorted(), .index(), .find(), .rfind()

What is the output of the following code?

```
>>> "Spam Ham Eggs".index('Spam', 1)
```

( ) Spam
( ) 0
( ) 1
(X) ValueError: substring not found
( ) TypeError: index() takes 1 argument (2 given)

Explanation:
index('Spam', 1) searches for 'Spam' beginning at 2nd character or index 1 (1st character is index 0). Substring not found and throws **ValueError**

Question >>>

***** .sort(), sorted(), .index(), .find(), .rfind()

What is the output of the following code?

```
t = "Spam Ham"
print(t.find("Ham", 0) == t.index("Ham", 0))
print(t.find("Eggs", 0) == t.index("Eggs", 0))
```

( ) True True
( ) True False
(X) True will be printed followed by ValueError: substring not found
( ) True will be printed followed by TypeError: find() takes 1 argument (2 given)

Explanation:
https://docs.python.org/3/library/stdtypes.html#str.find
https://docs.python.org/3/library/stdtypes.html#str.index
>>> t = "Spam Ham"
>>> t.find("Ham", 0)
5
>>> t.index("Ham", 0)
5
>>> t.find("Eggs", 0)
-1
>>> t.index("Eggs", 0)
ValueError: substring not found

Question >>>

***** .sort(), sorted(), .index(), .find(), .rfind()

What is the output of the following code?

```
t = "Spam Ham"
print(t.rfind("am") == t.find("am"))
print(t.rfind("am", 3) == t.find("am", 3))
print(t.rfind("am", -3) == t.find("am", -3))
```

( ) False False False
(X) False True True
( ) True True True
( ) True will be printed followed by TypeError: rfind takes 1 argument (2 given)

Explanation:
https://docs.python.org/3/library/stdtypes.html#str.rfind
https://docs.python.org/3/library/stdtypes.html#str.find
>>> t = "Spam Ham"
>>> t.rfind("am"), t.find("am") # Spam H**am**, Sp**am** Ham
(6, 2)
>>> t.rfind("am", 3), t.find("am", 3) # Spam H**am**, Spam H**am**
(6, 6)
>>> t.rfind("am", -3), t.find("am", -3) # Spam H**am**, Spam H**am**
(6, 6)

Question >>>

Exam block #4: Object-Oriented Programming (34%)

Objectives covered by the block (12 items)

* ideas: class, object, property, method, encapsulation, inheritance, grammar vs class, superclass, subclass

Which of the option(s) is valid given the code below?

```
class Spam:
    ''' This is class Spam '''
    pass
```

(X) The code compiles but will not output anything
( ) This is class Spam will be printed
( ) SyntaxError: invalid syntax
( ) The file should be saved as Spam.py

Explanation:
This is a minimal class declaration. The triple-quoted string functions as a comment.

Question >>>

What is the output of the following code?

```
def spam():
    class Ham:
        def eggs(self):
            print('Hello World')
    return Ham()

spam().eggs()
```

( ) No output
(X) Hello World
( ) SyntaxError: invalid syntax
( ) AttributeError: spam() has no attribute 'eggs'

Explanation:
class can be declared inside function definitions (def statements)
Question >>>

What is the output of the following code?

```
def spam():
    h = Ham()
    h.eggs()
    class Ham:
        def eggs(self):
            print('Hello World')
    return

spam()
```

( ) No output
( ) Hello World
( ) SyntaxError: invalid syntax
(X) UnboundLocalError: local variable 'Ham' referenced before assignment

Explanation:
Class definitions, like function definitions (def statements) must be executed before they have any effect.
Question >>>

***** ideas: class, object, property, method, encapsulation, inheritance, grammar vs class, superclass, subclass

What is the output of the following code?

```
class Foo:
    bar = 'spam'

f1 = Foo()
f2 = Foo()
f2.bar = 'ham'
Foo.bar = 'eggs'
print(f1.bar, f2.bar, Foo.bar)
```

( ) spam ham eggs
(X) eggs ham eggs
( ) eggs eggs eggs
( ) AttributeError: type object 'Foo' has no attribute 'bar'

Explanation:

```
class Foo:
    bar = 'spam'
f1 = Foo()
f2 = Foo()
>>> id(f1.bar),id(f2.bar),id(Foo.bar) # initial references are the same
(58327552, 58327552, 58327552)
f2.bar = 'ham' # updating the instance variable updates the instance variable reference
>>> id(f1.bar),id(f2.bar),id(Foo.bar)
(58327552, 58329472, 58327552)
Foo.bar = 'eggs' # updating the class variable updates all variable references not updated
>>> id(f1.bar),id(f2.bar),id(Foo.bar)
(52210080, 58329472, 52210080)
>>> f1.bar, f2.bar, Foo.bar
('eggs', 'ham', 'eggs')
```

Question >>>

***** ideas: class, object, property, method, encapsulation, inheritance, grammar vs class, superclass, subclass

What is the output of the following code?

```
class Spam:
    HAM = 1
    def __init__(self, v=2):
        self.v = v + Spam.HAM
        Spam.HAM += 1
a = Spam()
b = Spam(3)
print(a.v, b.v)
```

( ) TypeError: __init__() missing 1 required positional argument: 'v'

( ) 3 3
( ) 3 4
(X) 3 5

Explanation:
>>> Spam()
self.v=3 v=2 Spam.HAM=1
>>> # Spam.HAM += 1 accessed as class var and carried over next
>>> Spam(3)
self.v=5 v=3 Spam.HAM=2

Question >>>

What is the output of the following code?

```
class Ham:
    def __init__(self):
        self.v1 = 1
class Spam(Ham):
    def __init__(self):
        self.v2 = 2
s = Spam()
print(s.v1,s.v2)
```

( ) 0 2
( ) 1 2
( ) Invalid Syntax
(X) AttributeError: 'Spam' object has no attribute 'v1'

Explanation:
The parent's __init__ was never called hence v1 was never initialized and will not be visible.

Question >>>

***** ideas: class, object, property, method, encapsulation, inheritance, grammar vs class, superclass, subclass

What is the output of the following code?

```
class Ham:
    v = 1
    def v0(self):
```

```
        return self.v
class Spam(Ham):
    v = 2
s = Spam()
h = Ham()
print(s.v0(), h.v0())
```

( ) 1 1
(X) 2 1
( ) 2 2
( ) AttributeError: 'Spam' object has no attribute 'v0'

Explanation:

```
>>> class Ham:
...     v = 1
...     def v0(self):
...         return self.v
>>> class Spam(Ham):
...     v = 2
>>> s = Spam()
>>> h = Ham()
>>> Spam.__mro__  # method resolution order
(<class '__main__.Spam'>, <class '__main__.Ham'>, <class 'object'>)
>>> Ham.__mro__   # method resolution order
(<class '__main__.Ham'>, <class 'object'>)
>>> s.v0()       # returns 2 based on Spam.__mro__
2
>>> h.v0()       # returns 1 based on Ham.__mro__
1
```

Question >>>

What is the output of the following code?

```
>>> def foo(self, p):
...     print('Hello',p)
>>> class Spam:
...     bar = foo
>>> s = Spam()
```

```
>>> s.bar('World')
```

( ) No output
(X) Hello World
( ) SyntaxError: invalid syntax
( ) NameError: name 'foo' is not defined

Explanation:
Any function object that is a class attribute defines a method for instances of that class. It is not necessary that the function definition is textually enclosed in the class definition: assigning a function object to a local variable in the class is also ok.

Question >>>

***** ideas: class, object, property, method, encapsulation, inheritance, grammar vs class, superclass, subclass

What is the output of the following code?

```
1 class Spam:
2     def __init__(self, v):
3         self.ham = v
4         self.__ ham = self. ham + 1
5 s = Spam(100)
6 print(s.ham ,s.__ham)
```

( ) 100 101
( ) Error in Line 3
( ) Error in Line 4
( ) Error in Line 5
(X) Error in Line 6

Explanation:
Variables prefixed with __ are private and not accessible outside the class.

Question >>>

***** ideas: class, object, property, method, encapsulation, inheritance, grammar vs class, superclass, subclass

What is the output of the following code?

```
class A:
```

```
    def spam(self):
        return 'A.spam'
    def ham(self):
        return self.spam()
class B:
    def spam(self):
        return 'B.spam'
class C(B, A):
    pass

c = C()
print(c.spam(), c.ham())
```

( ) TypeError: Cannot create a consistent method resolution
(X) B.spam B.spam
( ) B.spam A.spam
( ) A.spam A.spam

Explanation:

https://docs.python.org/3/library/stdtypes.html#class.__mro__

```
>>> class A:
...     def spam(self):
...         return 'A.spam'
...     def ham(self):
...         return self.spam()
>>> class B:
...     def spam(self):
...         return 'B.spam'
>>> class C(B, A):
...     pass
>>> c = C()
>>> C.__mro__  # method resolution order for function execution
(<class '__main__.C'>, <class '__main__.B'>, <class '__main__.A'>,
<class 'object'>)
>>> c.spam.__qualname__ # prints B.spam
'B.spam'
>>> c.ham.__qualname__  # calls spam of B based on __mro__ prints
```

B.spam
'**A.ham**'
Question >>>

***** ideas: class, object, property, method, encapsulation, inheritance, grammar vs class, superclass, subclass
***** ideas: class, object, property, method, encapsulation, inheritance, grammar vs class, superclass, subclass

What is the output of the following code?

```
class Spam:
    def foo(self):
        print('Super Spam')
class Ham:
    def foo(self):
        print('Super Ham')
class Eggs(Spam, Ham):
    def foo(self):
        super().foo()
e = Eggs()
e.foo()
```

( ) No output
(X) Super Spam
( ) Super Ham
( ) Super Spam Super Ham
( ) Super Ham Super Spam

Explanation:
Search for attributes inherited from a parent class are depth-first, left-to-right. Attribute is searched in Eggs, if not found, it is searched in Spam, then (recursively) in the base classes of Spam, if not found, it is searched in Ham, and so on.

Question >>>

Which option(s) are valid replacements for the marked section below.

```
class Bar:
```

```
    def __init__(self):
        self.x = 1
class Foo(Bar):
    def __init__(self):
        <<< INSERT CODE HERE >>>
        self.y = 2
f = Foo()
print(f.x,f.y)
```

[ ] Blank. Code will work without replacement
[X] super(Spam, self).__init__ ()
[X] Bar.__init__ (self)
[ ] None. All results in AttributeError: 'Foo' object has no attribute 'x'

Explanation:
The Bar's __init__ was never called hence x was never initialized and will not be visible. print(f.x, f.y) results in AttributeError: 'Foo' object has no attribute 'x'

Question >>>

***** ideas: class, object, property, method, encapsulation, inheritance, grammar vs class, superclass, subclass

Select the choices which will return TRUE?

```
class X:
    pass
class Y:
    pass
class Z(X, Y):
    pass
```

[ ] issubclass(X, Z) and issubclass(Y, Z)
[X] issubclass(Z, X) and issubclass(Z, Y)
[X] issubclass(Z, (list, X, Y))
[ ] issubclass(Z, X, Y))

Explanation:
https://docs.python.org/3/library/functions.html#issubclass

Question >>>

What is the output of the following code?

```
class A(object): pass
class C(A,A): pass
```

( ) No output
( ) SyntaxError: invalid syntax
(X) TypeError: duplicate base class A
( ) NameError: name 'object' is not defined

Explanation:
Duplicate base class not allowed.

Question >>>

***** instance vs class variables: declaring, initializing

What is the output of the following code?

```
class MyClass:
    FOO = 100
    def __init__(self):
        self.bar = []
    def add(self, p):
        self.bar.append(p)

d, e = MyClass(), MyClass()
d.add('spam')
e.add('ham')
e.FOO = 200
MyClass.FOO = 300
print(d.bar, d.FOO, e.bar, e.FOO)
```

( ) ['spam'] 300 ['ham'] 300
(X) ['spam'] 300 ['ham'] 200
( ) ['spam'] 100 ['ham'] 200
( ) ['spam'] 100 ['ham'] 300

Explanation:
>>> class MyClass:

```
...     FOO = 100
...     def __init__(self):
...         self.bar = []
...     def add(self, p):
...         self.bar.append(p)
>>> d, e = MyClass(), MyClass()
>>> id(d.bar), id(e.bar)
(49644232, 6296232)
>>> d.add('spam')
>>> e.add('ham')
>>> e.FOO = 200
>>> id(d.FOO), id(e.FOO), id(MyClass.FOO)
(1736748512, 1736750112, 1736748512)
>>> MyClass.FOO = 300
>>> d.bar, d.FOO, e.bar, e.FOO
(['spam'], 300, ['ham'], 200)
```

Question >>>

Which of the following option(s) is valid given the code below?

```
1 class Spam:
2     HAM = 100
3     def __init__(self):
4         self.eggs = []
5     def add(self, p):
6         self.eggs.append(p)
```

[ ] HAM is an instance variable
[X] eggs is an instance variable
[X] HAM is a class variable
[ ] eggs is a class variable
[ ] Error in LINE 4

Explanation:
https://docs.python.org/3/tutorial/classes.html#class-and-instance-variables

Question >>>

***** instance vs class variables: declaring, initializing

What is the output of the following code?

```
1 class Spam:
2     ham = 0
3     def __init__(self):
4         ham = 100
5
6 s, t = Spam(), Spam()
7 s.ham, t.ham = 200, 300
8 Spam.ham = 500
9 print(s.ham, t.ham)
```

( ) 500 500
(X) 200 300
( ) Error in Line 2
( ) Error in Line 4
( ) Error in Line 8

Explanation:

```
>>> class Spam:
...     ham = 0
...     def __init__(self):
...         ham = 100
>>> s, t = Spam(), Spam()
>>> s.ham, t.ham = 200, 300
>>> id(s.ham), id(t.ham), id(Spam.ham)
(1642116128, 52376480, 1642112928)
>>> Spam.ham = 500
>>> id(s.ham), id(t.ham), id(Spam.ham)
(1642116128, 52376480, 52379568)
>>> s.ham, t.ham
(200, 300)
```

Question >>>

* __dict__ property (objects vs classes)

Select the option(s) which will return the dictionary or other mapping object used to store an object's (writable) attributes of the following code

```
class Person:
    name = "John"
```

```
    age = 36
    country = "USA"
p = Person()
```

[X] vars(Person)
[ ] vars(p)
[X] Person.__dict__
[ ] p.__dict__

Explanation:
https://docs.python.org/3/library/functions.html?#vars
Question >>>

***** private components (instance vs classes), name mangling

What is the output of the following code?

```
class Ham:
    def __init__(self):
        print(type(self).__name__ + '.__init__()', end=' ')
        self.__update()
    def update(self):
        print(type(self).__name__ + '.update()')
    __update = update

Ham()
```

( ) The script will run but will not output anything
( ) Ham.__init__()
(X) Ham.__init__() Ham.update()
( ) AttributeError: 'Ham' object has no attribute '_Ham__update'

Explanation:

```
>>> class Ham:
...     def __init__(self):  #2 initialize Ham
...         print(type(self).__name__ + '.__init__()', end=' ')  #3 print Ham.__init()
...         self.__update()  #4 call __update
...     def update(self):    #6 execute update via reference __update
...         print(type(self).__name__ + '.update()')  #7 print Ham.update()
```

```
...     __update = update    #5 save reference to __update
>>> Ham()                   #1 call initialize Ham
```

Question >>>

***** private components (instance vs classes), name mangling

Which of the option(s) below are valid calls given the code below?

```
>>> class Spam:
...     __ham = 0
...     def __eggs(self):
...         __ham = 100
...         return __ham
...     eggs = __eggs
>>> s = Spam()
<<< INSERT CODE HERE >>>
```

[X] >>> s.eggs()
[ ] >>> s.__eggs()
[X] >>> s._Spam__eggs()
[ ] >>> s.__ham
[X] >>> s._Spam__ham

Explanation:

```
>>> class Spam:
...     __ham = 0
...     def __eggs(self):
...         __ham = 100
...         return __ham
...     eggs = __eggs
>>> s = Spam()
>>> s.eggs()
100
>>> s.__eggs()
AttributeError: 'Spam' object has no attribute '__eggs'
>>> s._Spam__eggs()
100
>>> s.__ham
AttributeError: 'Spam' object has no attribute '__ham'
```

>>> s._Spam__ham
0

Question >>>

***** methods: declaring, using, self parameter

What is the output of the following code?

```
class Ham:
    def __init__(self):
        print(type(self).__name__ + '.__init__()', end=' ')
        self.update()
    def update(self):
        print(type(self).__name__ + '.update()', end=' ')
    def update(self, param):
        print(type(self).__name__ + '.update(param)', end=' ')
Ham()
```

( ) Ham.__init__() Ham.update()
( ) Ham.__init__() Ham.update(param)
( ) SyntaxError: invalid syntax
(X) TypeError: update() missing 1 required positional argument: 'param'

Explanation:
Python will disregard all except the last defined function with the same name within the same class.
>>> help(Ham.update)
Help on function update in module __main__:
update(self, param)

Question >>>

***** methods: declaring, using, self parameter

What is the output of the following code?

```
class Ham:
    def __init__(self):
        print(type(self).__name__ + '.__init__()', end=' ')
        self.__update()
    def update(self):
```

```
        print(type(self).__name__ + '.update()', end=' ')
    __update = update

class Spam(Ham):
    def update(self, param):
        print(type(self).__name__ + '.update(param)', end=' ')

Ham()
Spam()
```

( ) Ham.__init__() Ham.update() Ham.__init__() Ham.update()
(X) Ham.__init__() Ham.update() Spam.__init__() Spam.update()
( ) Ham.__init__() Ham.update() Spam.__init__() Spam.update(param)
( ) TypeError: update() missing 1 required positional argument: 'param'

Explanation:
Ham will save the reference of the function update to __update. __update() will reference the update defined in Ham for both Ham and Spam.

Question >>>

What is the output of the following code?

```
class Ham:
    def __init__(self):
        print(type(self).__name__ + '.__init__()', end=' ')
        self.update()
    def update(self):
        print(type(self).__name__ + '.update()', end=' ')

class Spam(Ham):
    def update(self, param):
        print(type(self).__name__ + '.update(param)', end=' ')

Ham()
Spam()
```

( ) Ham.__init__() Ham.update() Ham.__init__() Ham.update()
( ) Ham.__init__() Ham.update() Spam.__init__() Spam.update()
( ) Ham.__init__() Ham.update() Spam.__init__() Spam.update(param)
(X) TypeError: update() missing 1 required positional argument: 'param'

Explanation:

>>> Ham.__mro__
(<class '__main__.Ham'>, <class 'object'>)
>>> Spam.__mro__
(<class '__main__.Spam'>, <class '__main__.Ham'>, <class 'object'>)
__init__ for both Ham and Spam will call the update without the parameter. Using __mro__, it will fail on Spam because the update defined in Spam has 1 required positional argument param.

Question >>>

* methods: declaring, using, self parameter

Which of the statements below is valid?

```
class A:
    def __init__(self):
        pass
    def spam(self):
        pass
    def ham(self):
        return <CALL spam>
a = A()
<CALL ham>
```

[X] Replace **<CALL spam>** with self.spam()
[ ] Replace **<CALL spam>** with self.spam(self)
[X] Replace **<CALL ham>** with a.ham()
[ ] Replace **<CALL ham>** with a.ham(a)

Explanation:

Often, the first argument of a method is called self. This is nothing more than a convention: the name self has absolutely no special meaning to Python. Note, also, there is no need to supply the self argument with a parameter.

Question >>>

***** introspection: hasattr() (objects vs classes), __name__, __module__, __bases__ properties

What is the output of the following code?

```
>>> class Spam: pass
>>> hasattr(Spam(), 'ham')
```

(X) False
( ) AttributeError: 'Spam' object has no attribute 'ham'
( ) TypeError: hasattr(): attribute must be type
( ) NameError: name 'hasattr' is not defined
( ) SyntaxError: invalid syntax

Explanation:
https://docs.python.org/3/library/functions.html#hasattr

Question >>>

Select the choices which will return TRUE?

```
class Spam:
    ham = 36
spam = Spam()
```

[X] hasattr(spam, 'ham')
[X] hasattr(Spam, 'ham')
[ ] hasattr('Spam', 'ham')
[ ] hasattr('spam', 'ham')
[ ] spam.hasattr('ham')

Explanation:

```
>>> class Spam:
...   ham = 36
>>> spam = Spam()
>>> hasattr(spam, 'ham')
True
>>> hasattr(Spam, 'ham')
True
>>> hasattr('Spam',  'ham')
False
>>> hasattr('spam', 'ham')
False
```

>>> spam.hasattr('ham')
AttributeError: 'Spam' object has no attribute 'hasattr'
Question >>>

***** introspection: hasattr() (objects vs classes), __name__, __module__, __bases__ properties

Which of the option(s) below is/are valid given the following code?

```
class Spam:
    __ham, ham = '__ham', 'ham'
    def __eggs(self):
        pass
    eggs = __eggs

s = Spam()
<<< INSERT CODE HERE >>>
```

[X] print(Spam.__name__)
[ ] print(s.__name__)
[X] print(s._Spam__eggs.__name__)
[ ] print(s._Spam__ham.__name__)
[X] print(s.eggs.__name__)
[ ] print(s.ham.__name__)

Explanation:

```
>>> class Spam:
...     __ham, ham = '__ham', 'ham'
...     def __eggs(self):
...      pass
...     eggs = __eggs
>>> s = Spam()
>>> print(Spam.__name__)
```

Spam

```
>>> print(s.__name__)
```

AttributeError: 'Spam' object has no attribute '__name__'

```
>>> print(s._Spam__eggs.__name__)
```

__eggs

```
>>> print(s._Spam__ham.__name__)
```

AttributeError: 'str' object has no attribute '__name__'
>>> print(s.eggs.__name__)
__eggs
>>> print(s.ham.__name__)
AttributeError: 'str' object has no attribute '__name__'
Question >>>

***** introspection: hasattr() (objects vs classes), __name__, __module__, __bases__ properties

Which of the option(s) below are valid given the following code?

```
class Spam:
    __ham, ham = '__ham', 'ham'
    def __eggs(self):
        pass
    eggs = __eggs

s = Spam()
<<< INSERT CODE HERE >>>
```

[ ] print(__module__)
[X] print(Spam.__module__)
[X] print(s._Spam__eggs.__module__)
[X] print(s.eggs.__module__)
[X] print(s.__module__)
[ ] print(s.ham.__module__)

Explanation:
>>> class Spam:
... __ham, ham = '__ham', 'ham'
... def __eggs(self):
... pass
... eggs = __eggs
>>> s = Spam()
>>> __module__
NameError: name '__module__' is not defined
>>> Spam.__module__
'__main__'

>>> s._Spam__eggs.__module__
'__main__'
>>> s.eggs.__module__
'__main__'
>>> s.__module__
'__main__'
>>> s.ham.__module__
AttributeError: 'str' object has no attribute '__module__'
Question >>>

***** introspection: hasattr() (objects vs classes), __name__, __module__, __bases__ properties

Which of the option(s) below are valid given the following code?

```
class Spam:
    __ham, ham = '__ham', 'ham'
    def __eggs(self):
        pass
    eggs = __eggs

s = Spam()
<<< INSERT CODE HERE >>>
```

[ ] print(__bases__)
[X] print(Spam.__bases__)
[ ] print(s._Spam__eggs.__bases__)
[X] print(type(s.eggs).__bases__)
[X] print(type(s).__bases__)
[ ] print(s.ham.__bases__)

Explanation:

>>> class Spam:
... __ham, ham = '__ham', 'ham'
... def __eggs(self):
... pass
... eggs = __eggs
>>> s = Spam()
>>> __bases__

NameError: name '__bases__' is not defined
>>> Spam.__bases__
(<class 'object'>,)
>>> s._Spam__eggs.__bases__
AttributeError: 'function' object has no attribute '__bases__'
>>> type(s.eggs).__bases__
(<class 'object'>,)
>>> type(s).__bases__
(<class 'object'>,)
>>> s.ham.__bases__
AttributeError: 'str' object has no attribute '__bases__'
Question >>>

***** inheritance: single, multiple, isinstance(), overriding, not is and is operators

What is the output of the following code?

```
class Eggs:
    def __init__(self):
        print('Eggs', end=' ')
class Ham(Eggs):
    def __init__(self):
        print('Ham', end=' ')
class Spam(Ham):
    pass

s = Spam()
```

( ) No output
(X) Ham
( ) Eggs
( ) Ham Eggs
( ) TypeError: __init__() takes 1 positional argument but 0 were given

Explanation:
Spam() will executes the first __init__() it finds in the class method resolution order.
>>> class Eggs:

```
...     def __init__(self):
...         print('Eggs', end=' ')
>>> class Ham(Eggs):
...     def __init__(self):
...         print('Ham', end=' ')
>>> class Spam(Ham):
...     pass
>>> s = Spam()
```
Ham
```
>>> Spam.__mro__            # method resolution order
```
(<class '__main__.Spam'>, <class '__main__.Ham'>, <class '__main__.Eggs'>, <class 'object'>)
```
>>> s.__init__.__qualname__  # __init__ associated with Spam
```
'Ham.__init__'

Question >>>

***** inheritance: single, multiple, isinstance(), overriding, not is and is operators

What is the output of the following code?

```
class Ham:
    def __init__(self):
        print('Ham', end=' ')
class Eggs:
    def __init__(self, end=' '):
        print('Eggs')
class Spam(Ham, Eggs):
    pass
```

( ) No output
(X) Ham
( ) Eggs
( ) Ham Eggs
( ) TypeError: __init__ takes 1 positional argument but 0 were given

Explanation:
Spam() will executes the first __init__() it finds in the class method resolution order.

```
>>> class Ham:
...     def __init__(self):
...         print('Ham', end=' ')
>>> class Eggs:
...     def __init__(self, end=' '):
...         print('Eggs')
>>> class Spam(Ham, Eggs):
...     pass
>>> s = Spam()
```
Ham
```
>>> Spam.__mro__            # method resolution order
```
(<class '__main__.Spam'>, <class '__main__.Ham'>, <class '__main__.Eggs'>, <class 'object'>)
```
>>> s.__init__.__qualname__  # __init__ associated with Spam
```
'Ham.__init__'

Question >>>

***** inheritance: single, multiple, isinstance(), overriding, not is and is operators

Select the choices which will return TRUE?

```
>>> class X: pass
>>> class Y: pass
>>> class Z(X, Y): pass
>>> x, y, z = X(), Y(), Z()
```

[ ] isinstance(X, z) and isinstance(Y, z)
[X] isinstance(z, X) and isinstance(z, Y)
[X] isinstance(z, (list, X, Y))
[ ] isinstance((list, X, Y), z)
[ ] isinstance(z, X, Y)

Explanation:
https://docs.python.org/3/library/functions.html#isinstance

Question >>>

Which option will return True given the following code?

```
>>> class A(object): pass
```

```
>>> class B(object): pass
>>> class C(object): pass
>>> class D(object): pass
>>> class E(object): pass
>>> class K1(A,B,C): pass
>>> class K2(D,B,E): pass
>>> class K3(D,A):   pass
>>> k = K3()
```

[X] >>> isinstance(k, K3)
[X] >>> isinstance(k, D)
[X] >>> isinstance(k, (list, K2, K3))
[ ] >>> isinstance(k, (list, K1, K2))
[X] >>> isinstance(k, (list, A, B, C, D, E))

Explanation:
>>> isinstance(k, K3)
True
>>> isinstance(k, D)
True
>>> isinstance(k, (list, K2, K3))
True
>>> isinstance(k, (list, K1, K2))
False
>>> isinstance(k, (list, A, B, C, D, E))
True
Question >>>

***** inheritance: single, multiple, isinstance(), overriding, not is and is operators

Select which option contains the correct function name for the following generator?

```
class Spam:
    def <<Replace 1>>(self, p=""):
        self.s = p
        self.i = 0
```

```
    def <<Replace 2>>(self):
        return self
    def <<Replace 3>>(self):
        if self.i == len(self.s):
            raise StopIteration
        v = self.s[self.i]
        self.i += 1
        return v
```

(X) 1=__init__, 2=__iter__, 3=__next__
( ) 1=__init__, 2=__iterator__, 3=__next__
( ) 1=__init__, 2=__iterate__, 3=__next__
( ) 1=__init__, 2=__pop__, 3=__push__
( ) 1=__init__, 2=__generator__, 3=__next

Explanation:
https://docs.python.org/3/library/stdtypes.html#generator-types

Question >>>

***** inheritance: single, multiple, isinstance(), overriding, not is and is operators

***** constructors: declaring and invoking

Select the choices to invoke the constructor of Spam and assign the instance to s

```
class Spam:
    def __init__(self, v=0):
        self.ham = v + 1
```

[X] s = Spam()
[X] s = Spam(10)
[ ] s = Spam(s, 10)
[ ] s = Spam.__init__(s)
[ ] AttributeError: 'Spam' object has no attribute 'ham'

Explanation:

```
>>> class Spam:
...     def __init__(self, v=0):
...         self.ham = v + 1
```

```
>>> s = Spam()    # valid
>>> s = Spam(10)  # valid
>>> del s         # unset previous assignment
>>> s = Spam(s, 10)
```
NameError: name 's' is not defined
```
>>> del s         # unset previous assignment
>>> s = Spam.__init__(s)
```
NameError: name 's' is not defined

Question >>>

* constructors: declaring and invoking

How do you instantiate class Spam of the code below?

```
class Spam:
    def __init__(self):
        self.bar = 0
```

( ) You can't because there's an AttributeError in the code
(X) spam = Spam()
( ) spam = Spam(None)
( ) spam = Spam(Spam)

Explanation:
Often, the first argument of a method is called self. This is nothing more than a convention: the name self has absolutely no special meaning to Python. Note, also, there is no need to supply the self argument with a parameter.

Question >>>

What is the output of the following code?

```
class Spam:
    def __init__(self, v):
        self.ham = v + 1
spam = Spam(1)
print(spam.ham)
```

( ) AttributeError: 'Spam' object has no attribute 'ham'
( ) TypeError: __init__() takes 2 positional arguments but 1 were given
( ) 1
(X) 2

Explanation:
Often, the first argument of a method is called self. This is nothing more than a convention: the name self has absolutely no special meaning to Python. Note, also, there is no need to supply the self argument with a parameter.
Question >>>

* polymorphism
* __name__, __module__, __bases__ properties, __str__() method

Select the line number from the options(s) which will print **Spam**

```
1 class Spam:
2     def v0(self):
3           print(__name__)
4 print(__name__)
5 s = Spam()
6 s.v0()
7 print(s.__class__.__name__)
8 print(Spam.__name__)
9 print(s.__name__)
```

[ ] Line 3
[ ] Line 4
[X] Line 7
[X] Line 8
[ ] Line 9

Explanation:

```
>>> class Spam:
...     def v0(self):
...         print(__name__)
>>> print(__name__)
__main__
>>> s = Spam()
>>> s.v0()
__main__
>>> print(s.__class__.__name__)
```

Spam
>>> print(Spam.__name__)
Spam
>>> print(s.__name__)
AttributeError: 'Spam' object has no attribute '__name__'
Question >>>

What is the output of the following code?

```
class X:
    def spam(): pass
ham = X.spam
print(ham.__name__)
```

( ) SyntaxError: invalid syntax
( ) X.spam
(X) spam
( ) ham

Explanation:
https://docs.python.org/3/library/stdtypes.html#definition.__name__
>>> class X:
...   def spam(): pass
>>> ham = X.spam
>>> ham.__qualname__
'X.spam'
>>> ham.__name__
'spam'
Question >>>

***** __name__, __module__, __bases__ properties, __str__() method

What is the output of the following code?

```
F=type('Food',(),{'remember2buy':'spam'})
E=type('Eggs',(F,),{'remember2buy':'eggs'})
G=type('GoodFood',(E,F),{})
print(F.__name__, E.__name__, G.__name__)
```

( ) No output

( ) SyntaxError: invalid syntax
( ) F E G
(X) Food Eggs GoodFood
( ) AttributeError: type object 'Food' has no attribute '__name__'

Explanation:
https://www.python.org/download/releases/2.3/mro/

Question >>>

***** __name__, __module__, __bases__ properties, __str__() method
***** __name__, __module__, __bases__ properties, __str__() method
***** __name__, __module__, __bases__ properties, __str__() method

What is the output of the following code?

```
class Ham:
    def __str__(self): return "Ham"

class Spam(Ham): pass

print(Spam())
```

( ) No output
( ) <__main__.Spam object at 0x03100FD0>
(X) Ham
( ) TypeError: __str__() missing 1 required positional argument

Explanation:
https://docs.python.org/3/reference/datamodel.html?#object.__str__

Question >>>

Which option will print **Spam** given the following code?

```
>>> class Spam:
...     def __str__(self): return "Spam"
>>> s = Spam()
```

[ ] >>> s

[X] >>> print(s)
[ ] >>> Spam()
[X] >>> s.__str__()
[ ] >>> s.__repr__()

Explanation:

>>> s
<__main__.Spam object at 0x03AB3F88>
>>> print(s)
Spam
>>> Spam()
<__main__.Spam object at 0x03AB3FD0>
>>> s.__str__()
'Spam'
>>> s.__repr__()
'<__main__.Spam object at 0x03AB3F88>'

Question >>>

***** multiple inheritance, diamonds

What is the output of the following code?

```
F=type('Food',(),{'remember2buy':'spam'})
E=type('Eggs',(F,),{'remember2buy':'eggs'})
G=type('GoodFood',(E,F),{})
print(F.remember2buy, E.remember2buy, G.remember2buy)
```

( ) SyntaxError: invalid syntax
( ) No Output
( ) Food Eggs GoodFood
( ) spam eggs
(X) spam eggs eggs

Explanation:

https://www.python.org/download/releases/2.3/mro/

Question >>>

***** multiple inheritance, diamonds

Which of the option(s) below are valid given the following code?

```
O = object
```

```
class X(O): pass
class Y(O): pass
class A(X,Y): pass
class B(Y,X): pass
<<< INSERT CODE HERE >>>
```

[ ] class Foo(A, B): pass
[ ] class Foo(B, A): pass
[X] class Foo(A, X): pass
[ ] class Foo(X, A): pass
[X] class Foo(B, Y): pass
[ ] class Foo(Y, B): pass

Explanation:
https://www.python.org/download/releases/2.3/mro/

Question >>>

Which of the option(s) below are valid given the following code?

```
O = object
class F(O): pass
class E(O): pass
class D(O): pass
class C(D,F): pass
class B(D,E): pass
class A(B,C): pass
<<< INSERT CODE HERE >>>
```

[X] class Foo(A, B): pass
[ ] class Foo(B, A): pass
[X] class Foo(A, C): pass
[ ] class Foo(C, A): pass
[X] class Foo(B, C): pass
[X] class Foo(C, B): pass
[X] class Foo(C, B, E): pass
[ ] class Foo(E, B, C): pass

Explanation:
https://www.python.org/download/releases/2.3/mro/

Question >>>

Which of the options below are valid given the following code?

```
class A(object): pass
class B(object): pass
class C(object): pass
class D(object): pass
class E(object): pass
class K1(A,B,C): pass
class K2(D,B,E): pass
class K3(D,A):   pass
<<< INSERT CODE HERE >>>
```

[X] class Foo(K1,K2,K3): pass
[X] class Foo(K1,K3,K2): pass
[X] class Foo(K2,K1,K3): pass
[X] class Foo(K2,K3,K1): pass
[X] class Foo(K3,K1,K2): pass
[X] class Foo(K3,K2,K1): pass

Explanation:

https://www.python.org/download/releases/2.3/mro/

Question >>>

Exam block #5: Miscellaneous (List Comprehensions, Lambdas, Closures, and I/O Operations) (22%)

Objectives covered by the block (9 items)

* list comprehension: if operator, using list comprehensions

What is the output of the following code?

```
>>> [[c for c in range(r)] for r in range(3) if r != 0]
```

(X) [[0], [0, 1]]
( ) [[1], [1, 2]]
( ) [[0], [1]]
( ) [[1], [2]]
( ) SyntaxError: invalid syntax

Explanation:
>>> [[c for c in range(r)] for r in range(3) if r != 0]
[[0], [0, 1]]
>>> [[c for c in range(r)] for r in [0, 1, 2] if r != 0]
[[0], [0, 1]]
>>> [[c for c in range(r)] for r in [1, 2]]
[[0], [0, 1]]
>>> [c for c in [[0], [0, 1]]]
[[0], [0, 1]]

Question >>>

What is the output of the following code?

```
>>> [_ for _ in range(10) if not _%2 ]
```

(X) [0, 2, 4, 6, 8]
( ) [2, 4, 6, 8, 10]
( ) [1, 3, 5, 7, 9]
( ) [0, 1, 3, 4, 5, 6, 7, 8, 9]
( ) SyntaxError: invalid syntax

Explanation:
>>> [_ for _ in range(10) if not _%2]
[0, 2, 4, 6, 8]
>>> # odd numbers are removed. not _%2 are numbers without remainder when divided by 2
>>> [_ for _ in [0, 1, 2, 3, 4, 5, 6, 7, 8, 9] if not _%2]
[0, 2, 4, 6, 8]
>>> [_ for _ in [0, 2, 4, 6, 8]]
[0, 2, 4, 6, 8]
Question >>>

***** list comprehension: if operator, using list comprehensions

What is the output of the following code?

```
>>> [False for i in range(3)]
```

( ) []
( ) [False]
( ) [0, 1, 2]
(X) [False, False, False]
( ) SyntaxError: invalid syntax

Explanation:
Same as add **False** 3 times in the list.
Question >>>

What is the output of the following code?

```
>>> [i for i in range(1)][-1]
```

( ) SyntaxError: invalid syntax
( ) []
( ) [0]
(X) 0
( ) None

Explanation:
>>> [i for i in range(1)]
[0]
>>> [0][-1]
0

Question >>>

What is the output of the following code?

```
>>> len([[c for c in range(r)] for r in range(3)])
```

( ) 2
(X) 3
( ) TypeError: len() takes exactly one argument (2 given)
( ) SyntaxError: invalid syntax

Explanation:

>>> [[c for c in range(r)] for r in range(3)]
[[], [0], [0, 1]]
>>> [[c for c in range(r)] for r in [0, 1, 2]]
[[], [0], [0, 1]]
>>> [c for c in [[], [0], [0, 1]]]
[[], [0], [0, 1]]

Question >>>

Which option will produce a non-empty list?

[X] lst = [i for i in range(1, 5)]
[ ] lst = [i for i in range(5, 1)]
[ ] lst = [i for i in range(-1, -5)]
[X] lst = [i for i in range(-5, -1)]
[ ] lst = [i for i in range(0, -5)]

Explanation:

>>> [i for i in range(1, 5)]
[1, 2, 3, 4]
>>> [i for i in range(5, 1)]
[]
>>> [i for i in range(-1, -5)]
[]
>>> [i for i in range(-5, -1)]
[-5, -4, -3, -2]
>>> [i for i in range(0, -5)]
[]

Question >>>

What is the output of the following code?

```
spam = [x * x for x in range(5)]
del spam[spam[2]]
print(spam)
```

(X) [0, 1, 4, 9]
( ) IndexError: list index out of range
( ) TypeError: range() takes exactly 2 arguments (1 given)
( ) SyntaxError: invalid syntax

Explanation:

>>> spam = [x * x for x in range(5)]
>>> spam # [0*0, 1*1, 2*2, 3*3, 4*4]
[0, 1, 4, 9, 16]
>>> spam[spam[2]] # spam[spam[2] = 4] == spam[4]
16
>>> del spam[spam[2]] # removes [0, 1, 4, 9, (**16) <- remove**]

Question >>>

What is the output of the following code?

```
x = [_ for _ in range(10)]
del x[0:-2]
print(x)
```

(X) [8, 9]
( ) [9, 8]
( ) [0, 1, 2, 3, 4, 5, 6, 7]
( ) [7, 6, 5, 4, 3, 2, 1, 0]
( ) [2, 3, 4, 5, 6, 7, 8, 9]
( ) [9, 8, 7, 6, 5, 4, 3, 2]

Explanation:

>>> x = [_ for _ in range(10)]
>>> x
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> x[0:-2]
[0, 1, 2, 3, 4, 5, 6, 7]
>>> del x[0:-2] # [**(0, 1, 2, 3, 4, 5, 6, 7)** <- remove, 8, 9]

Question >>>

What is the output of the following code?

```
>>> [i // i for i in range(0,3)]
```

( ) [0, 1, 1]
( ) [1, 1]
( ) [0, 1, 2]
(X) ZeroDivisionError: integer division or modulo by zero

Explanation:
>>> [i for i in range(0,3)]
[0, 1, 2]
>>> [i // i for i in range(0,3)] # [0//0 <- Error, 1//1, 2//2]
ZeroDivisionError: integer division or modulo by zero

Question >>>

What is the output of the following code?

```
>>> [2 ** x for x in range(5)]
```

( ) [0, 2, 4, 6, 8]
(X) [1, 2, 4, 8, 16]
( ) [2, 4, 6, 8, 10]
( ) SyntaxError: invalid syntax

Explanation:
>>> [x for x in range(5)]
[0, 1, 2, 3, 4]
>>> [2 ** x for x in range(5)] # [2**0, 2**1, 2**2, 2**3, 2**4]
[1, 2, 4, 8, 16]

Question >>>

What is the output of the following code?

```
spam = [[x for x in range(4)] for y in range(4)]
for r in range(4):
    for c in range(4):
        spam[r][c] += 5
print(spam)
```

( ) [[5, 6, 7, 8], [5, 6, 7, 8]]
(X) [[5, 6, 7, 8], [5, 6, 7, 8], [5, 6, 7, 8], [5, 6, 7, 8]]
( ) Invalid Syntax

( ) TypeError: 'list' object does not support item assignment

Explanation:

```
>>> spam = [[x for x in range(4)] for y in range(4)]
>>> spam
[[0, 1, 2, 3], [0, 1, 2, 3], [0, 1, 2, 3], [0, 1, 2, 3]]
>>> for r in range(4): # This loops iterates over the elements
...     for c in range(4): # and adds 5 to all values
...         spam[r][c] += 5
>>> spam
[[5, 6, 7, 8], [5, 6, 7, 8], [5, 6, 7, 8], [5, 6, 7, 8]]
```

Question >>>

How many stars will the following code print?

```
l = [[i for i in range(2)] for i in range(2)]
for i in range(2):
    if l[0][i] % l[1][i] == 0:
        print('*')
```

(X) 0
( ) 1
( ) 2
( ) 4

Explanation:

```
>>> [[i for i in range(2)] for i in range(2)]
[[0, 1], [0, 1]]
>>> l[0][0] % l[1][0] # 0 % 0 results ZeroDivisionError no stars printed
ZeroDivisionError: integer division or modulo by zero
```

Question >>>

***** lambdas: defining and using lambdas, self-defined functions taking lambda as as arguments; map(), filter();

Which option is a valid definition of a lambda assigned to f that adds the parameter x and y?

[X] f = lambda x, y : x + y
[ ] f = lambda (x, y):(x + y)
[ ] f = lambda (x, y): x + y

[X] f = lambda x, y : (x + y)

Explanation:

https://docs.python.org/3/reference/expressions.html#lambda

>>> (lambda x, y : x + y)(10, 20)

30

>>> (lambda (x, y):(x + y))(10, 20)

SyntaxError: invalid syntax

>>> (lambda (x, y): x + y)(10, 20)

SyntaxError: invalid syntax

>>> (lambda x, y : (x + y))(10, 20)

30

Question >>>

What is the output of the following code?

```
f = lambda x: 10
print(f(20))
```

( ) 0
(X) 10
( ) 20
( ) SyntaxError: invalid syntax

Explanation:

Lambda expression returns 10 no matter what parameter is passed

Question >>>

What is the output of the following code?

```
func = lambda x: return x
print(func(10))
```

( ) No output
( ) 10
( ) NameError: name 'x' is not defined
(X) SyntaxError: invalid syntax

Explanation:

Lambda expects an expression after the parameter list. return is a statement not an expression.

Question >>>

What is the output of the following code?

```
>>> (lambda x: assert x != 2)(2)
```

( ) No output
( ) AssertionError
( ) NameError: name 'x' is not defined
(X) SyntaxError: invalid syntax

Explanation:
Lambda expects an expression after the parameter list. assert is a statement not an expression.

Question >>>

What is the output of the following code?

```
>>> a, b = 10, 20
>>> (lambda: b, lambda: a)[a < b]()
```

(X) 10
( ) 20
( ) TypeError: tuple indices must be integers or slices
( ) Invalid Syntax

Explanation:
(lambda: b, lambda: a) or (20, 10) is a tuple. The first item can be accessed by index 0 or False and the 2nd item with 1 or True.

```
>>> a, b = 10, 20
>>> (lambda: b)(),(lambda: a)()
(20, 10)
>>> (20, 10)[False]
20
>>> (20, 10)[True]
10
>>> (20, 10)[a < b]
10
```

Question >>>

***** lambdas: defining and using lambdas, self-defined functions taking lambda as as arguments; map(), filter();

What is the output of the following code?

```
spam = lambda x, f: x + f(x)
```

```
print(spam(2, lambda x: x * x), end=' ')
print(spam(2, lambda x: x + 3))
```

( ) SyntaxError: invalid syntax
(X) 6 7
( ) 4 5
( ) 6 10

Explanation:

```
>>> (lambda x: x * x)(2)  # x=2 * x=2 = 4
4
>>> (lambda x: x + 3)(2)  # x=2 + 3 = 5
5
>>> 2 + 4, 2 + 5         # x=2 + f(2)=4 == 6, x=2 + f(2)=5 == 7
(6, 7)
```

Question >>>

***** lambdas: defining and using lambdas, self-defined functions taking lambda as as arguments; map(), filter();

What is the output of the following code?

```
>>> list(map(lambda x: x*2, range(3)))
```

( ) [0, 1, 2]
(X) [0, 2, 4]
( ) [[0,0], [1,2], [2,4]]
( ) SyntaxError: invalid syntax

Explanation:

https://docs.python.org/3/library/functions.html#map

```
>>> list(map(lambda x: x*2, range(3))) # range(3) = [0, 1, 2]
[0, 2, 4]
>>> # map lambda x*2 to every item in the iterable = [0*2, 1*2, 2*2]
>>> list(map(lambda x: x*2, [0, 1, 2]))
[0, 2, 4]
>>> list([0, 2, 4])
[0, 2, 4]
```

Question >>>

What is the output of the following code?

```
>>> list(map(lambda x: x+10, [1, 2, 3]))
```

(X) [11, 12, 13]
( ) [11, 12, 13, 1, 2, 3]
( ) [[11, 12, 13],[1, 2, 3]]
( ) [{11:1},{12:2},{13:3}]
( ) [{11:[1, 2, 3]},{12:[1, 2, 3]},{13:[1, 2, 3]}]
( ) Invalid Syntax

Explanation:
https://docs.python.org/3/library/functions.html#map
>>> # map lambda x+10 to every item in the iterable = [1+10, 2+10, 3+10]
>>> list(map(lambda x: x+10, [1, 2, 3]))
[11, 12, 13]
>>> list([11, 12, 13])
[11, 12, 13]
Question >>>

***** lambdas: defining and using lambdas, self-defined functions taking lambda as as arguments; map(), filter()

What is the output of the following code?

```
>>> list(filter(lambda x: x%2, [1, 2, 3, 4, 5, 6, 7, 8, 9]))
```

( ) [1, 2, 3, 4, 5, 6, 7, 8, 9]
(X) [1, 3, 5, 7, 9]
( ) [2, 4, 6, 8]
( ) [1, 0, 1, 0, 1, 0, 1, 0, 1]
( ) [0, 1, 0, 1, 0, 1, 0, 1, 0]

Explanation:
https://docs.python.org/3/library/functions.html#filter
>>> # check lambda x%2 to every item in the iterable
>>> # [1%2, 2%2, 3%2, 4%2, 5%2, 6%2, 7%2, 8%2, 9%2]
>>> # remove items the evaluates to False or 0
>>> # [1, 0, 1, 0, 1, 0, 1, 0, 1]
>>> list(filter(lambda x: x%2, [1, 2, 3, 4, 5, 6, 7, 8, 9]))
[1, 3, 5, 7, 9]
Question >>>

***** closures: meaning, defining, and using closures

Which option is True about closures?

[X] closures is always nested inside a function
[ ] closures can be defined outside a function
[X] closures have access to a free variable in outer scope
[ ] closures have no access to variables in outer scope
[X] closures have access to global variables
[X] closures can define and modify nonlocal variables
[ ] closures are not allowed to define nonlocal variables

Explanation:
A closure is a nested function which has access to a free variable from an enclosing function that has finished its execution. Three characteristics of a Python closure are:

- it is a nested function
- it has access to a free variable in outer scope
- it is returned from the enclosing function

A free variable is a variable that is not bound in the local scope. In order for closures to work with immutable variables such as numbers and strings, we have to use the nonlocal keyword.
Python closures help avoiding the usage of global values and provide some form of data hiding but can access them. They are used in Python decorators.
Question >>>

* closures: meaning, defining, and using closures

What is the output of the following code?

```
x = 'x'
def main():
    y = 'y'
    def spam():
        print(x, y, z)
        return
    spam()
z = 'z'
if __name__ == "__main__":
    main()
```

( ) No output
(X) x y z

( ) NameError: name 'z' is not defined
( ) SyntaxError: invalid syntax

Explanation:

```
>>> x = 'x'                  #1 x = 'x'
>>> def main():                 #5 main()
...     y = 'y'               #6 y = 'y'
...     def spam():            #8 spam()
...         print(x, y, z)     #9 print x, y, z
...         return            #10 return
...     spam()                 #7 execute spam()
>>> z = 'z'                   #2 z = 'z'
>>> if __name__ == "__main__":  #3 evaluates to True
...     main()                 #4 execute main()
```

Question >>>

* closures: meaning, defining, and using closures

What is the output of the following code?

```
def spam(x):
    def ham(y):
        return x * y
    return ham
f = spam(2)
print(f(3))
```

( ) SyntaxError: invalid syntax
( ) 0
(X) 6
( ) NameError: name 'y' is not defined

Explanation:

```
1 | def spam(x):
5 | f = spam(2)
>>> Call to spam, line 1
..... x = 2
1 | def spam(x):
2 |     def ham(y):
4 |     return ham
```

<<< Return value from spam: ham
5 | f = spam(2)
..... f = ham
6 | print(f(3))
>>> Call to ham in line 5
..... y = 3
..... x = 2
2 | def ham(y):
3 | return x * y
<<< Return value from ham: 6
6

Question >>>

What is the output of the following code?

```
def spam(x):
    y = 2
    def ham(z):
        return x + y + z
    return ham

for i in range(3):
    eggs = spam(i)
    print(eggs(i+3), end=' ')
```

(X) 5 7 9
( ) 6 8 10
( ) 5 6 7
( ) SyntaxError: invalid syntax
( ) NameError: name z is not defined

Question >>>

What is the output of the following code?

```
def spam(x):
    y = 2
    return lambda z: x + y + z
```

```
for i in range(3):
    eggs = spam(i)
    print(eggs(i+3), end=' ')
```

(X) 5 7 9
( ) 6 8 10
( ) 5 6 7
( ) SyntaxError: invalid syntax
( ) NameError: name z is not defined

Question >>>

What is the output of the following code?

```
def main():
    x = 100
    a = [x, 200, 300]
    def spam():
        x = 500
        a[0] = x
        return
    spam()
    print(x, a)
if __name__ == "__main__":
    main()
```

( ) 100 [100, 200, 300]
(X) 100 [500, 200, 300]
( ) 500 [500, 200, 300]
( ) SyntaxError: invalid syntax
( ) NameError: name x is not defined

Explanation:

1 | def main():
2 | x = 100
3 | a = [x, 200, 300]
..... **a = [100, 200, 300]**
..... **len(a) = 3**
4 | def spam():

```
8 |     spam()
>>> Call to spam, line 4
..... a = [100, 200, 300]
..... len(a) = 3
4 |     def spam():
5 |         x = 500
6 |         a[0] = x
..... a = [500, 200, 300]
7 |         return
<<< Return value from spam: None
8 |     spam()
..... a = [500, 200, 300]
9 |     print(x, a)
100 [500, 200, 300]
```

Question >>>

***** I/O Operations: I/O modes, predefined streams, handles; text/binary modes
open(), errno and its values; close()
.read(), .write(), .readline(); readlines() (along with bytearray())

Select the choice(s) which is TRUE?

```
1 import sys
2 temp = sys.stdout
3 sys.stdout = open('spam.txt', 'w')
4 print("Hello World")
5 sys.stdout.close()
6 sys.stdout = temp
7 print("Good Bye")
```

[ ] An empty file 'spam.txt' will be created and the screen will display the text "Hello World" and "Good Bye"

[X] A file 'spam.txt' containing "Hello World" will be created and the screen will display the text "Good Bye"

[ ] A file 'spam.txt' containing "Hello World" will be created and the screen will display the text "Hello World" and "Good Bye"

[ ] No file will be created and the screen will display the text "Hello World"

and "Good Bye"
[ ] io.UnsupportedOperation in line 3
Explanation:
```
>>> import sys
>>> temp = sys.stdout
>>> sys.stdout = open('spam.txt', 'w') # assign ALL print to write on spam.txt
>>> print("Hello World")              # write Hello World on spam.txt
>>> sys.stdout.close()
>>> sys.stdout = temp                 # revert ALL print back to screen
>>> print("Good Bye")                 # print Good Bye on screen
```
Question >>>

What will open("spam.txt", "rt") return?
( ) Numeric status code
( ) The entire content of 'spam.txt'
( ) String filename
(X) File object
( ) TypeError: open() argument 2 must be int, not str
Explanation:
https://docs.python.org/3/library/functions.html?#open
Question >>>

***** I/O Operations: I/O modes, predefined streams, handles; text/binary modes
open(), errno and its values; close()
.read(), .write(), .readline(); readlines() (along with bytearray())

How do you use the BytesIO class in the io package imported on the following code?

```
import io
```

( ) spam = BytesIO()
(X) spam = io.BytesIO()
( ) spam = io->BytesIO()
( ) spam = io:BytesIO()
( ) spam = io::BytesIO()
Explanation:
https://docs.python.org/3/library/io.html#io.BytesIO

Question >>>

What functions can you call to read the Buffered Stream on the following code?

```
import io
spam = io.BytesIO()
spam.write("Hello, world!".encode('ascii'))
ham = spam.getbuffer()
spam.seek(0)
```

[X] spam.read()
[X] spam.read1()
[X] spam.readinto(ham)
[X] spam.readinto1(ham)
[ ] spam.read(ham)

Explanation:
https://docs.python.org/3/library/io.html#buffered-streams

Question >>>

Choose all correct ways to create a bytearray bar?

[ ] bar = b'confuse the cat'
[X] bar = bytearray()
[X] bar = bytearray(range(10))
[ ] bar = bytearray('confuse the cat')
[X] bar = bytearray(b'confuse the cat')

Explanation:
https://docs.python.org/3/library/stdtypes.html#bytearray

Question >>>

* I/O Operations: I/O modes, predefined streams, handles; text/binary modes
open(), errno and its values; close()
.read(), .write(), .readline(); readlines() (along with bytearray())
* I/O Operations: I/O modes, predefined streams, handles; text/binary modes
open(), errno and its values; close()
.read(), .write(), .readline(); readlines() (along with bytearray())

What is the output of the code?

```
for spam in open('spam.txt', 'rt'):
    print(spam, end='')
```

given spam.txt

```
This is LINE 1
This is LINE 2
```

( ) Nothing is printed
( ) This is LINE 1 is printed in an infinite loop
( ) This is LINE 1 This is LINE 2 is printed in a single line
(X) This is LINE 1 and This is LINE 2 is printed in separate lines
( ) TypeError: 'TextIOWrapper' object is not iterable

Explanation:
Prints **This is LINE 1** and **This is LINE 2** in separate lines because there is a newline character that was read in the file.

Question >>>

What are the valid access modes available for the open() function?

[X] r, w, a
[X] rb, wb, ab
[X] r+, w+, a+
[X] rb+, wb+, ab+
[ ] br, bw, ba
[ ] br+, bw+, ba+
[X] r+b, w+b, a+b

Explanation:
https://docs.python.org/3/library/functions.html?#open

Question >>>

***** I/O Operations: I/O modes, predefined streams, handles; text/binary modes
open(), errno and its values; close()
.read(), .write(), .readline(); readlines() (along with bytearray())

Which option(s) are valid results of the following code if spam.txt does not exist?

```
import sys, errno
```

```
try:
    open("spam.txt", "r")
except:
    if sys.exc_info()[1].errno == errno.ENOENT:
        print(errno.errorcode[sys.exc_info()[1].errno])
```

[ ] No output
[ ] 2
[X] ENOENT
[ ] No such file or directory
[ ] Error Line 3
[ ] Error Line 5
[ ] Error Line 6

Explanation:
'r' open for reading (default), will raise errno.**ENOENT** or **No such file or directory** if the file does not exist.

Question >>>

Which option(s) are valid results of the following code if spam.txt does not exist?

```
import sys, errno
try:
    open("spam.txt", "x")
except:
    if sys.exc_info()[1].errno == errno.ENOENT:
        print(errno.errorcode[sys.exc_info()[1].errno])
```

[X] No output
[ ] 2
[ ] ENOENT
[ ] No such file or directory
[ ] Error Line 3
[ ] Error Line 5
[ ] Error Line 6

Explanation:
'x' open for exclusive creation, will not fail since spam.txt does not exist.

Question >>>

Which option(s) are valid results of the following code if spam.txt exist?

```
import sys
try:
    open("spam.txt", "x")
except:
    print(sys.exc_info()[1].errno)
```

[ ] No output
[ ] invalid mode: 'x'
[X] [Errno 17] File exists: 'spam.txt'
[ ] Error Line 3
[ ] Error Line 5

Explanation:
'x' open for exclusive creation, failing if the file already exists
Question >>>

***** I/O Operations: I/O modes, predefined streams, handles; text/binary modes
open(), errno and its values; close()
.read(), .write(), .readline(); readlines() (along with bytearray())

Which of the option(s) below are valid calls given the code below?

```
file = open('spam.txt', 'r+')
file.close()
<<< INSERT CODE HERE >>>
```

[ ] No file operations are allowed after close()
[X] file.close()
[ ] file.read()
[ ] file.readline()
[ ] file.write()

Explanation:
>>> file = open('spam.txt', 'r+')
>>> file.close()
>>> file.close()
>>> file.read()
ValueError: I/O operation on closed file.

>>> file.readline()
ValueError: I/O operation on closed file.
>>> file.write('X')
ValueError: I/O operation on closed file.
Question >>>

***** I/O Operations: I/O modes, predefined streams, handles; text/binary modes
open(), errno and its values; close()
.read(), .write(), .readline(); readlines() (along with bytearray())

What is the output of the code?

```
spam = open("spam.txt", "r")
print(spam.read(2))
```

given spam.txt

```
This is LINE 1
This is LINE 2
This is LINE 3
```

(X) The first 2 characters **Th** is printed
( ) The first 2 lines **This is LINE 1** and **This is LINE 2** is printed
( ) The first 2 characters **Th** is skipped **is is LINE 1** is printed
( ) The 2nd line **This is LINE 2** is printed

Explanation:
https://docs.python.org/3/library/io.html#io.RawIOBase.read
Question >>>

***** I/O Operations: I/O modes, predefined streams, handles; text/binary modes
open(), errno and its values; close()
.read(), .write(), .readline(); readlines() (along with bytearray())

Which option(s) will write "Hello World" on an open file f?
[X] f.write("Hello World")
[ ] f.writeln("Hello World")
[ ] f.writeline("Hello World")
[X] f.writelines("Hello World")

[X] f.writelines(["Hello World"])
[ ] f.print("Hello World")

Explanation:
https://docs.python.org/3/library/io.html#i-o-base-classes

Question >>>

What is the result of the following code?

```
with open('spam.txt', 'r+') as file:
    line = file.read()
file.write(line)
```

given spam.txt

```
12345
```

[X] spam.txt will still contain 12345
[ ] spam.txt will now contain 123451
[ ] spam.txt will now contain 1234512345
[ ] NameError: name 'line' is not defined
[X] ValueError: I/O operation on closed file.

Explanation:
https://docs.python.org/3/tutorial/inputoutput.html#reading-and-writing-files

Question >>>

***** I/O Operations: I/O modes, predefined streams, handles; text/binary modes
open(), errno and its values; close()
.read(), .write(), .readline(); readlines() (along with bytearray())

Which option(s) are valid read on an open file f?

[X] f.read()
[X] f.read(1)
[ ] f.readln()
[X] f.readline()
[X] f.readlines()
[X] f.readlines(1)

Explanation:
https://docs.python.org/3/library/io.html#i-o-base-classes

Question >>>

What is the output of the code?

```
spam = open("spam.txt", "r")
print(spam.readline(2))
```

given spam.txt

```
This is LINE 1
This is LINE 2
This is LINE 3
```

(X) The first 2 characters **Th** is printed
( ) The first 2 lines **This is LINE 1** and **This is LINE 2** is printed
( ) The first 2 characters **Th** is skipped **is is LINE 1** is printed
( ) The 2nd line **This is LINE 2** is printed

Explanation:
https://docs.python.org/3/library/io.html#io.IOBase.readline

Question >>>

***** I/O Operations: I/O modes, predefined streams, handles; text/binary modes
open(), errno and its values; close()
.read(), .write(), .readline(); readlines() (along with bytearray())

What is the output of the following code?

```
>>> b'the quick brown fox'.translate(None, b'aeiou')
```

( ) b'aeiou'
(X) b'th qck brwn fx'
( ) aeiou
( ) th qck brwn fx
( ) the quick brown fox

Explanation:
https://docs.python.org/3/library/stdtypes.html#bytearray.translate

bytearray.**translate**(*table*, */*, *delete=b''*)
Return a copy of the bytes or bytearray object where all bytes occurring in the optional argument delete are removed, and the remaining bytes have been mapped through the given translation table, which must be a bytes object of

length 256.

You can use the bytes.maketrans() method to create a translation table.

Set the *table* argument to None for translations that only delete characters

Question >>>

Exam block #6: Bonus questions

Python 3.9.4 REPL can be launched by which of the following option given the output of py -0?

```
C:\Users\jlacanienta>py -0
Installed Pythons found by py Launcher for Windows
-3.9-64 *
```

[X] py
[X] py -3
[X] py -3.9
[X] py -3.9-64
[ ] py -3.9.4
[ ] py -3.9.4-64

Explanation:

```
C:\Users\jlacanienta>py
Python 3.9.4 (tags/v3.9.4:1f2e308, Apr  6 2021, 13:40:21) [MSC v
bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more informati
>>> exit()
C:\Users\jlacanienta>py -3
Python 3.9.4 (tags/v3.9.4:1f2e308, Apr  6 2021, 13:40:21) [MSC v
bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more informati
>>> exit()
C:\Users\jlacanienta>py -3.9-64
Python 3.9.4 (tags/v3.9.4:1f2e308, Apr  6 2021, 13:40:21) [MSC v
bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more informati
>>> exit()
C:\Users\jlacanienta>py -3.9.4
Unknown option: -3
```

```
usage:
C:\Users\jlacanienta\AppData\Local\Programs\Python\Python39\p
[option] ... [-c cmd | -m mod | file | -] [arg] ...
Try `python -h' for more information.
C:\Users\jlacanienta>py -3.9.4-64
Unknown option: -3
usage:
C:\Users\jlacanienta\AppData\Local\Programs\Python\Python39\p
[option] ... [-c cmd | -m mod | file | -] [arg] ...
Try `python -h' for more information.
```

Question >>>

What is the output of the following code?

```
a, b = 10, 20
print(a < b and a or b)
```

(X) 10
( ) 20
( ) True
( ) Invalid Syntax

Explanation:
https://docs.python.org/3/reference/expressions.html#operator-precedence
>>> a, b = 10, 20
>>> a < b and a or b
10
>>> # 10 < 20 and 10 or 20
>>> # True and 10 or 20
>>> # 10 or 20

Question >>>

Select all valid variable names

[ ] is
[X] then
[ ] elif
[ ] pass

[X] catch
[X] exception
Explanation:
Keywords not allowed to be used as variable names
>>> keyword.kwlist
['False', 'None', 'True', '__peg_parser__', 'and', 'as', 'assert', 'async', 'await', 'break', 'class', 'continue', 'def', 'del', 'elif', 'else', 'except', 'finally', 'for', 'from', 'global', 'if', 'import', 'in', 'is', 'lambda', 'nonlocal', 'not', 'or', 'pass', 'raise', 'return', 'try', 'while', 'with', 'yield']
Question >>>

Select the values considered false.
[X] None
[X] False
[X] zero of any numeric type (0, 0L, 0.0, 0j)
[X] empty sequence ('', (), [])
[X] class with a __nonzero__()definition
[X] class with __len__() that returns integer 0
Explanation:
https://docs.python.org/3/library/stdtypes.html#truth-value-testing
Question >>>

Select all valid bitwise operators
[X] <<
[X] >>
[X] &
[X] |
[X] ~
[X] ^
Explanation:
https://wiki.python.org/moin/BitwiseOperators
Question >>>

Which option(s) results in 12.34?
[ ] >>> 1234e2
[X] >>> 1234e-2
[X] >>> .1234e2

[ ] >>> .1234e-2
[ ] None

Explanation:

>>> 1234e2
123400.0
>>> 1234e-2
12.34
>>> .1234e2
12.34
>>> .1234e-2
0.001234

Question >>>

Select valid integer assignment for the variable spam?

[ ] spam = 1e0
[X] spam = 0b1
[X] spam = 0o1
[X] spam = 0x1
[ ] spam = \u0031 #The Unicode of 1 is U+0031

Explanation:

>>> type(1e0)
<class 'float'>
>>> type(0b1)
<class 'int'>
>>> type(0o1)
<class 'int'>
>>> type(0x1)
<class 'int'>
>>> type(\u0031)
SyntaxError: unexpected character after line continuation character

Question >>>

What is the output of the following code?

```
>>> ['spam', 'ham'] * 2
```

( ) SyntaxError: invalid syntax
( ) TypeError: unsupported operand type(s) for *: 'list' and 'int'

( ) ['spamspam', 'hamham']
(X) ['spam', 'ham', 'spam', 'ham']

Explanation:

List multiplication will duplicate the content by *n* times.

>>> ['spam'] * 2
['spam', 'spam']
>>> ['spam', 'ham'] * 2
['spam', 'ham', 'spam', 'ham']

Question >>>

What is the output of the following code?

```
>>> ('spam', ) * 2
```

( ) SyntaxError: invalid syntax
( ) TypeError: unsupported operand type(s) for *: 'tuple' and 'int'
( ) 'spamspam'
( ) ('spamspam')
(X) ('spam', 'spam')

Explanation:

Tuple multiplication will duplicate the content by *n* times.

>>> ('spam',) * 2
('spam', 'spam')
>>> ('spam', 'ham') * 2
('spam', 'ham', 'spam', 'ham')

Question >>>

What is the output of the following code?

```
>>> 1 - 2 / 3 // 4 + 5
```

( ) 4.0
( ) 5.0
(X) 6.0
( ) 7.0

Explanation:

https://docs.python.org/3/reference/expressions.html#operator-precedence
>>> 1 - 2 / 3 // 4 + 5 #1 2/3=0.67
6.0

>>> 1 - 0.67 // 4 + 5   #2 0.67/4=0.0
6.0
>>> 1 - 0.0 + 5
6.0
Question >>>

What is the output of the following code?

```
>>> 1 // 2 + 1 / 2
```

( ) 0.0
(X) 0.5
( ) 0.75
( ) 1.0

Explanation:
https://docs.python.org/3/reference/expressions.html#operator-precedence
>>> 1 // 2 + 1 / 2 #1 1//2=0
0.5
>>> 0 + 1 / 2   #2 1/2=0.5
0.5
>>> 0 + 0.5
0.5
Question >>>

What is the output of the following code?

```
>>> 1. /( 4. % 2.)
```

( ) 0.5
( ) 0.0
( ) Syntax Error
(X) ZeroDivisionError

Explanation:
https://docs.python.org/3/reference/expressions.html#operator-precedence
>>> 1. /(4. % 2.) #1 4.%2.=0.0
ZeroDivisionError: float division by zero
>>> 1. / (0.0)
ZeroDivisionError: float division by zero
Question >>>

What is the output of the following code?

```
x = 3
while x > 0:
    print(x, end='')
    x //= 2
```

( ) 3
(X) 31
( ) 31.50
( ) Infinite loop

Explanation:

1 | x = 3
2 | while x > 0:
3 | print(x, end='')
3
4 | x //= 2
..... **x = 1**
2 | while x > 0:
3 | print(x, end='')
1
4 | x //= 2
..... **x = 0**
2 | while x > 0:

Question >>>

What is the output of the following code?

```
>>> -1 // 2
```

(X) -1
( ) -1.0
( ) -0.5
( ) 0
( ) 0.5
( ) 1.0
( ) 1

Explanation:
// (floor division) operators yield the quotient of their arguments. The numeric arguments are first converted to a common type. floor division of integers results in an integer; the result is that of mathematical division with the 'floor' function applied to the result. Division by zero raises the ZeroDivisionError exception.

Question >>>

What is the output of the following code?

```
a, b = 0, 1
print(a ^ a, a ^ b, b ^ a, b ^ b)
```

( ) 0 1 1 1
(X) 0 1 1 0
( ) 1 0 0 1
( ) 1 0 0 0

Explanation:
The ^ operator yields the bitwise XOR (exclusive OR) of its arguments, which must be integers. Results 1 if arguments are not both 0 or not both 1.

Question >>>

What is the output of the following code?

```
t = True
f = not t
t = t or f
f = t and f
t, f = f, t
print(t, f)
```

( ) False False
(X) False True
( ) True False
( ) True True
( ) Syntax Error

Explanation:
t = True #1 t=True f=NA
f = not t #2 t=True f=False

t = t or f #3 t=True f=False
f = t and f #4 t=True f=False
t, f = f, t #5 t=False t=True
print(t, f) #6 prints **False True**

Question >>>

Select options which will print True based on the following code?

```
spam = True
```

[ ] print(spam = True)
[X] print(spam == True)
[ ] print(spam === True)
[X] print(spam is True)

Explanation:

>>> spam = True
>>> print(spam = True)
TypeError: 'spam' is an invalid keyword argument for print()
>>> print(spam == True)
True
>>> print(spam === True)
SyntaxError: invalid syntax
>>> print(spam is True)
True

Question >>>

Select options which will print True based on the following code?

```
spam = 1
```

[X] print(spam != 0)
[ ] print(spam !== 0)
[ ] print(spam !=== 0)
[ ] print(spam <> 0)
[X] print(spam is not 0)

Explanation:

>>> spam = 1
>>> print(spam != 0)
True

>>> print(spam !== 0)
SyntaxError: invalid syntax
>>> print(spam !=== 0)
SyntaxError: invalid syntax
>>> print(spam <> 0)
SyntaxError: invalid syntax
>>> print(spam is not 0)
True

Question >>>

What is the output of the following code?

```
a = 10
b = 20
c = b < a > 0 or a > b and b > a or a < b
print(c)
```

( ) SyntaxError: invalid syntax
(X) True
( ) False
( ) 1
( ) 0

Explanation:
https://docs.python.org/3/reference/expressions.html#operator-precedence
>>> a, b = 10, 20
>>> b < a > 0 or a > b and b > a or a < b
True
>>> # 20 < 10 > 0 or 10 > 20 and 20 > 10 or 10 < 20
>>> # True > 0 or False and True or True
>>> # True or False or True

Question >>>

Select all keyword argument of print()

[X] sep
[X] end
[X] file
[X] flush
[ ] format

Explanation:
https://docs.python.org/3/library/functions.html#print
Question >>>

What is the output of the following code?

```
1 print(int(10.10), end= " ")
2 print(int("10", 10), end= " ")
3 print(int("10", base=10), end= " ")
4 print(int(0o12), end= " ")
5 print(int(10))
```

(X) 10 10 10 10 10
( ) TypeError: int() takes at most 1 argument (2 given)
( ) TypeError: 'base' is an invalid keyword argument for int()
( ) Invalid syntax on Line 4 (0o12)

Explanation:
https://docs.python.org/3/library/functions.html#int
>>> int(10.10)
10
>>> int("10", 10)
10
>>> int("10", base=10)
10
>>> int(0o12)
10
>>> int(10)
10

Question >>>

What is the output of the code?

```
1 print(float('+1.23'), end=" ")
2 print(float(' -12345\n'), end=" ")
3 print(float('1e-003'), end=" ")
4 print(float('+1E6'), end=" ")
5 print(float('-Infinity'))
```

(X) 1.23 -12345.0 0.001 1000000.0 -inf
( ) ValueError: could not convert string to float in Line 1
( ) ValueError: could not convert string to float in Line 2
( ) ValueError: could not convert string to float in Line 3
( ) ValueError: could not convert string to float in Line 4
( ) ValueError: could not convert string to float in Line 5

Explanation:

https://docs.python.org/3/library/functions.html#float

>>> float('+1.23')
1.23
>>> float(' -12345\n')
-12345.0
>>> float('1e-003')
0.001
>>> float('+1E6')
1000000.0
>>> float('-Infinity')
-inf

Question >>>

What will call to output x/y/z*?

```
x, y, z = "x", "y", "z"
s = [x, y, z]
t = x, y, z
```

[X] print(x, y, z, sep='/', end="*\n")
[ ] print(s, sep='/', end="*\n")
[ ] print(t, sep='/', end="*\n")
[X] print('/'.join(s) + '*\n')
[ ] Syntax Error

Explanation:

>>> print(x, y, z, sep='/', end="*\n")
x/y/z*
>>> print(s, sep='/', end="*\n")
['x', 'y', 'z']*
>>> print(t, sep='/', end="*\n")

('x', 'y', 'z')*
>>> print('/'.join(s) + '*\n')
x/y/z*

Question >>>

What is the output of the following code?

```
>>> spam = ('S', 'P', 'A', 'M')
>>> s, p, _, _ = spam
>>> s
'S'
>>> p
'P'
>>> _
```

( ) No output
( ) 'A'
(X) 'M'
( ) 'AM'

Explanation:
"_" is a special variable in most Python REPLs that represents the result of the last expression evaluated by the interpreter.
e.g.
>>> _, _ = ('A', 'M')
>>> _
'M'

Question >>>

What is the output of the following code?

```
>>> for i in range(10):
...     pass
>>> i
```

( ) NameError: name 'i' is not defined
( ) 0
(X) 9
( ) 10

Explanation:
https://docs.python.org/3/library/stdtypes.html#range
Question >>>

What is the output of the following code?

```
total = 0
for i in range(1, 4) :
    i += 2
    total += i
else:
    total += 100
print(total)
```

(X) 112
( ) 12
( ) 108
( ) 8
( ) SyntaxError: invalid syntax

Explanation:
1 | total = 0
2 | for i in range(1, 4) :
..... **i = 1**
3 | i += 2
..... **i = 3**
4 | total += i
..... **total = 3**
2 | for i in range(1, 4) :
..... **i = 2**
3 | i += 2
..... **i = 4**
4 | total += i
..... **total = 7**
2 | for i in range(1, 4) :
..... i = 3
3 | i += 2

..... **i = 5**
4 | total += i
..... **total = 12**
2 | for i in range(1, 4) :
6 | total += 100
..... **total = 112**
7 | print(total)
112

Question >>>

What is the output of the following code?

```
for i in range(1, 4, 2):
    print(i, end=' ')
```

( ) TypeError: range expected at most 2 arguments, got 3
(X) 1 3
( ) 1, 4, 2
( ) 1 2 3 4

Explanation:
https://docs.python.org/3/library/stdtypes.html#range
1 | for i in range(1, 4, 2):
..... **i = 1**
2 | print(i, end=' ')
1
1 | for i in range(1, 4, 2):
..... **i = 3**
2 | print(i, end=' ')
3

Question >>>

What is the output of the following code?

```
x = { 'x': 1, 'y': 2}
for e in x:
    print(e, type(e), end=' ')
```

(X) x <class 'str'> y <class 'str'>
( ) 1 <class 'int'> 2 <class 'int'>

( ) ('x', 1) <class 'tuple'> ('y', 2) <class 'tuple'>
( ) 'x':1 <class 'iterable'> 'y':2 <class 'iterable'>

Explanation:

Iterating a dictionary calls __iter__() which iterates over the keys of a dictionary. e.g. Dictionary x contains the keys 'x' and 'y' that are both <class 'str'>

Question >>>

What is the output of the following code?

```
i = 0
total = 0
while i < 4 :
    i += 2
    total += i
else:
    total += 100
print(total)
```

( ) 4
( ) 6
( ) 104
(X) 106
( ) SyntaxError: invalid syntax

Explanation:

1 | i = 0
2 | total = 0
3 | while i < 4 :
4 | i += 2
..... **i = 2**
4 | total += i
..... **total = 2**
3 | while i < 4 :
4 | i += 2
..... **i = 4**
5 | total += i
..... **total = 6**

3 | while i < 4 :
7 | total += 100
..... **total = 106**
8 | print(total)
106

Question >>>

What is the output of the following code?

```
spam = [1, 2]
for i in range(2):
    spam.insert(-1, spam[i])
    print(spam, end=' ')
```

(X) [1, 1, 2] [1, 1, 1, 2]
( ) [1, 1, 2] [1, 1, 2, 2]
( ) [-1, 1, 2] [-1, -1, 1, 2]
( ) [1, 2, -1] [1, 2, -1, -1]
( ) [1, -1, 2] [1, -1, -1, 2]

Explanation:
https://docs.python.org/3/tutorial/datastructures.html?highlight=dictionary#more-on-lists
ist.**insert**(i, *x*)
Insert an item at a given position. The first argument is the index of the element before which to insert, so a.insert(0, x) inserts at the front of the list, and a.insert(len(a), x) is equivalent to a.append(x).

Question >>>

Which statement is CORRECT about the code below?

```
class Spam: pass
del Spam
ham = "Fubar"
del ham
ham = 'FuBar'
del ham[0]
ham = ["spam", "ham"]
```

```
del ham[1]
```

( ) No error
( ) del Spam is an error
( ) del ham is an error
(X) del ham[0] is an error
( ) del ham[1] is an error

Explanation:
Strings are immutable sequences of Unicode code points. Modification in any of its parts is not allowed.

```
>>> del Spam
>>> ham = "Fubar"
>>> del ham
>>> ham = 'FuBar'
>>> del ham[0]
TypeError: 'str' object doesn't support item deletion
>>> ham = ["spam", "ham"]
>>> del ham[1]
```

Question >>>

What is the output of the following code?

```
spam = [1, 2, 3]
ham = spam
del ham[:]
print(spam)
```

( ) [1, 2, 3]
(X) []
( ) Can't delete list
( ) SyntaxError: invalid syntax

Explanation:

```
>>> spam = [1, 2, 3]
>>> ham = spam
>>> ham[:]
[1, 2, 3]
>>> del ham[:]  # same as del ham[0:3]
>>> ham
```

[]
>>> spam
[]

Question >>>

Which option will result in the output **[1, 2, 3] [1, 2, 3] False**?

```
ham = [1, 2, 3]
<<< INSERT CODE HERE >>>
print(spam, ham, id(spam)==id(ham))
```

[ ] spam = ham
[X] spam = ham.copy()
[X] spam = ham[:]
[X] spam = list(ham)

Explanation:

>>> spam = ham; spam, ham, id(spam)==id(ham)
([1, 2, 3], [1, 2, 3], True)
>>> spam = ham.copy(); spam, ham, id(spam)==id(ham)
([1, 2, 3], [1, 2, 3], False)
>>> spam = ham[:]; spam, ham, id(spam)==id(ham)
([1, 2, 3], [1, 2, 3], False)
>>> spam = list(ham); spam, ham, id(spam)==id(ham)
([1, 2, 3], [1, 2, 3], False)

Question >>>

Which option will print "the quick brown fox" given the code below?

```
spam = ("the", "quick", "brown", "fox")
```

[X] print(spam[0],spam[1],spam[2],spam[3])
[ ] print(spam[1],spam[2],spam[3],spam[4])
[ ] print(spam[0],spam[-1],spam[-2],spam[-3])
[X] print(spam[-4],spam[-3],spam[-2],spam[-1])

Explanation:

>>> spam = ("the", "quick", "brown", "fox")
>>> print(spam[0],spam[1],spam[2],spam[3])
the quick brown fox
>>> print(spam[1],spam[2],spam[3],spam[4])

IndexError: tuple index out of range
>>> print(spam[0],spam[-1],spam[-2],spam[-3])
the fox brown quick
>>> print(spam[-4],spam[-3],spam[-2],spam[-1])
the quick brown fox
Question >>>

What is the output of the following code?

```
a, b = 10, 20
print((b, a) [a < b])
```

(X) 10
( ) 20
( ) TypeError: tuple indices must be integers or slices
( ) Invalid Syntax
Explanation:
(b, a) or (20, 10) is a tuple. The first item can be accessed by index 0 or False and the 2nd item with 1 or True.
>>> a, b = 10, 20
>>> t = (b, a)
>>> t[False]
20
>>> t[True]
10
>>> t[a < b]
10
Question >>>

Which option will create a tuple ham equal to ("brown", "fox") using the code below?

```
spam = ("the", "quick", "brown", "fox")
```

[X] ham = spam[2:]
[X] ham = spam[2:4]
[X] ham = spam[-2:]
[ ] ham = spam[-2:0]
[ ] ham = spam[-2:-1]

Explanation:
>>> spam = ("the", "quick", "brown", "fox")
>>> ham = spam[2:]; ham
('brown', 'fox')
>>> ham = spam[2:4]; ham
('brown', 'fox')
>>> ham = spam[-2:]; ham
('brown', 'fox')
>>> ham = spam[-2:0]; ham
()
>>> ham = spam[-2:-1]; ham
('brown',)
Question >>>

Which option will create a tuple ham equal to (0, 3, 6, 9) using the code below?

```
spam = (0, 1, 2, 3, 4, 5, 6, 7, 8, 9)
```

[X] ham = spam[::3]
[ ] ham = spam[0:3:9]
[X] ham = spam[0::3]
[ ] ham = spam[0:9:3]
[X] ham = spam[0:10:3]

Explanation:
>>> spam = (0, 1, 2, 3, 4, 5, 6, 7, 8, 9)
>>> ham = spam[::3]; ham
(0, 3, 6, 9)
>>> ham = spam[0:3:9]; ham
(0,)
>>> ham = spam[0::3]; ham
(0, 3, 6, 9)
>>> ham = spam[0:9:3]; ham
(0, 3, 6)
>>> ham = spam[0:10:3]; ham
(0, 3, 6, 9)
Question >>>

Which option is a valid way to create a tuple named spam?

[X] spam = ()
[ ] spam = ("the")
[X] spam = ("the",)
[X] spam = ("the", "quick", "brown", "fox")

Explanation:

>>> type(())
<class 'tuple'>
>>> type(("the"))
<class 'str'>
>>> type(("the",))
<class 'tuple'>
>>> type(("the", "quick", "brown", "fox"))
<class 'tuple'>

Question >>>

What is the output of the following code?

```
1 spam = ('0', 1, 2, 3, 4, 5, 6, 7, 8, 9)
2 spam[0] = 0
3 print(spam)
```

( ) ('0', 1, 2, 3, 4, 5, 6, 7, 8, 9)
( ) (0, 1, 2, 3, 4, 5, 6, 7, 8, 9)
( ) Error in Line 1
(X) Error in Line 2
( ) Error in Line 3

Explanation:

TypeError: 'tuple' object does not support item assignment

Question >>>

What is the output of the following code?

```
spam = {}
spam[1] = [1, 2]
spam[2] = [3, 4]
print(type(spam))
```

( ) <class 'list'>

( ) <class 'tuple'>
( ) <class 'set'>
(X) <class 'dict'>

Explanation:

https://docs.python.org/3/library/stdtypes.html#mapping-types-dict

Question >>>

What is the output of the following code?

```
a, b = 10, 20
print({True: a, False: b} [a < b])
```

(X) 10
( ) 20
( ) KeyError: True
( ) Invalid Syntax

Explanation:

{True: a, False: b} is a dictionary with True, False keys with corresponding a=10 and b=20 values.

```
>>> a, b = 10, 20
>>> d = {True: a, False: b}
>>> d[True]
10
>>> d[False]
20
>>> d[a < b]  # a < b == True
10
```

Question >>>

What is the output of the following code?

```
spam = { 'z':'x', 'x':'y', 'y':'z' }
ham = 'x'
for x in range(len(spam)):
    ham = spam[ham]
    print(ham, end="")
```

( ) xyz
( ) yz

(X) yzx
( ) zxy
( ) KeyError

Explanation:

1 | spam = { 'z':'x', 'x':'y', 'y':'z' }
..... **spam = {'z': 'x', 'x': 'y', 'y': 'z'}**
..... **len(spam) = 3**
2 | ham = 'x'
3 | for x in range(len(spam)):
..... **x = 0**
4 | ham = spam[ham]
..... **ham = 'y'**
5 | print(ham, end="")
y
3 | for x in range(len(spam)):
..... **x = 1**
4 | ham = spam[ham]
..... **ham = 'z'**
5 | print(ham, end="")
z
3 | for x in range(len(spam)):
..... **x = 2**
4 | ham = spam[ham]
..... **ham = 'x'**
5 | print(ham, end="")
x

[Question >>>]

Select all options which will return False given the code below?

```
tel = {'rick': 123, 'morty': 456}
```

[X] 'spam' in tel
[ ] tel['spam'] is not None
[X] tel.get('spam') is not None
[ ] tel.get('spam', True) is not None
[ ] tel.key('spam') is not None

Explanation:
>>> tel = {'rick': 123, 'morty': 456}
>>> 'spam' in tel
False
>>> tel['spam'] is not None
KeyError: 'spam'
>>> tel.get('spam') is not None
False
>>> tel.get('spam', True) is not None
True
>>> tel.key('spam') is not None
AttributeError: 'dict' object has no attribute 'key'
Question >>>

What is the output of the following code?

```
spam = {}
spam['f1'] = {'b1':11, 'b2':12}
spam['f2'] = {'b1':21, 'b2':22}
for ham in spam.keys():
    print(ham, end=' ')
```

(X) f1 f2
( ) b1 b2
( ) f1 b1 b2 f2 b1 b2
( ) f1 f2 b1 b2
( ) TypeError: keys() takes exactly 1 argument (0 given)
Explanation:
https://docs.python.org/3/library/stdtypes.html#dict.keys
>>> spam = {}
>>> spam['f1'] = {'b1':11, 'b2':12} # add f1 key
>>> spam.keys()
dict_keys(['f1'])
>>> spam['f2'] = {'b1':21, 'b2':22} # add f2 key
>>> spam.keys()
dict_keys(['f1', 'f2'])
Question >>>

Select all options that will print the key and value pair of the spam of the following code?

```
spam = {'b1':11, 'b2':12}
```

[X] for x in spam.items():print(x[0], x[1])
[X] for x, y in spam.items():print(x, y)
[ ] for x in spam.values():print(x[0], x[1])
[ ] for x, y in spam.values():print(x, y)
[ ] for x in spam:print(x[0], x[1])

Explanation:

```
>>> spam = {'b1':11, 'b2':12}
>>> for x in spam.items():print(x[0], x[1])
b1 11
b2 12
>>> for x, y in spam.items():print(x, y)
b1 11
b2 12
>>> for x in spam.values():print(x[0], x[1])
TypeError: 'int' object is not subscriptable
>>> for x, y in spam.values():print(x, y)
TypeError: cannot unpack non-iterable int object
>>> for x in spam:print(x[0], x[1])
b 1
b 2
```

Question >>>

What is the output of the following code?

```
>>> {False: 'No', 0: 'Nay', 0.0: 'Nope'}
```

( ) {False: 'No', 0: 'Nay', 0.0: 'Nope'}
( ) {False: 'No'}
(X) {False: 'Nope'}
( ) {False: 'No', 0: 'Nope'}
( ) {False: 'No', 0: 'Nay'}
( ) {0.0: 'No'}
( ) {0.0: 'Nope'}

Explanation:

Python treats False, 0, and 0.0 as the same keys and retains the key as False succeeding value overrides the previous one.

Question >>>

How will you call the functions **spam(), ham(), and eggs()** so you will output **spam ham eggs**?

```
def spam():
    print("spam", end=" ")
    def ham():
        print("ham", end=" ")
        def eggs():
            print("eggs")
            return None
        return eggs
    return ham
```

[ ] spam(); ham(); eggs()
[X] spam()()()
[X] x = spam(); y = x(); z = y()
[ ] spam().ham().eggs()
[ ] Invalid Syntax

Explanation:

```
>>> def spam():
...     print("spam", end=" ")
...     def ham():
...         print("ham", end=" ")
...         def eggs():
...             print("eggs")
...             return None
...         return eggs
...     return ham
>>> spam(); ham(); eggs()
NameError: name 'ham' is not defined
>>> spam()()()
spam ham eggs
```

>>> x = spam(); y = x(); z = y()
spam ham eggs
>>> spam().ham().eggs()
AttributeError: 'function' object has no attribute 'ham'
Question >>>

How many 'b's will be printed based on the output of the following code?

```
def spam(n):
    result = 'b'
    for i in range(n):
        result += result
        yield result
for s in spam(2):
    print(s, end="")
```

( ) SyntaxError: invalid syntax
( ) 0
( ) 2
( ) 4
(X) 6

Explanation:
Total of 6 **b** or **bbbbbb** printed. **bb** is printed in the first iteration and **bbbb** is printed in the 2nd iteration.
1 | def spam(n):
6 | for s in spam(2):
>>> Start generator spam, line 1
..... **n = 2**
1 | def spam(n):
2 | result = 'b'
3 | for i in range(n):
..... **i = 0**
4 | result += result
..... **result = 'bb'**
5 | yield result
<<< Yield value from spam: 'bb'
6 | for s in spam(2):

..... **s = 'bb'**
7 | print(s, end="")
bb
6 | for s in spam(2):
>>> Re-enter generator spam, line 5
..... **n = 2**
..... **result = 'bb'**
..... **i = 0**
5 | yield result
3 | for i in range(n):
..... **i = 1**
4 | result += result
..... **result = 'bbbb'**
5 | yield result
<<< Yield value from spam: 'bbbb'
6 | for s in spam(2):
7 | print(s, end="")
bbbb
6 | for s in spam(2):
>>> Re-enter generator spam, line 5
..... **n = 2**
..... **result = 'bbbb'**
..... **i = 1**
5 | yield result
3 | for i in range(n):

Question >>>

What is the output of the following code?

```
def foo():
    for x in range(5):
        yield x*x
for x in foo():
    print(x, end=" ")
```

( ) 0
( ) 0 1 2 3 4

(X) 0 1 4 9 16
( ) TypeError: 'int' object is not iterable

Explanation:

4 | for x in foo():
>>> Start generator foo
1 | def foo():
2 | for x in range(5):
..... **x = 0**
3 | yield x*x
<<< Yield value from foo: **0**
4 | for x in foo():
..... **x = 0**
5 | print(x, end=" ")
0
4 | for x in foo():
>>> Re-enter generator foo
..... **x = 0**
3 | yield x*x
2 | for x in range(5):
..... **x = 1**
3 | yield x*x
<<< Yield value from foo: **1**
4 | for x in foo():
..... **x = 1**
5 | print(x, end=" ")
1
4 | for x in foo():
>>> Re-enter generator foo
..... **x = 1**
3 | yield x*x
2 | for x in range(5):
..... **x = 2**
3 | yield x*x
<<< Yield value from foo: **4**
4 | for x in foo():
..... **x = 4**

5 | print(x, end=" ")
4
4 | for x in foo():
>>> Re-enter generator foo
..... **x = 2**
3 | yield x*x
2 | for x in range(5):
..... **x = 3**
3 | yield x*x
<<< Yield value from foo: **9**
4 | for x in foo():
..... **x = 9**
5 | print(x, end=" ")
9
4 | for x in foo():
>>> Re-enter generator foo
..... **x = 3**
3 | yield x*x
2 | for x in range(5):
..... **x = 4**
3 | yield x*x
<<< Yield value from foo: **16**
4 | for x in foo():
..... **x = 16**
5 | print(x, end=" ")
16

Question >>>

What is the output of the following code?

```
def spam(d, k, v):
    d[k]=v
    print(d, end=' ')
print(spam({}, '0', 'value'))
```

( ) SyntaxError: invalid syntax
( ) {}

( ) {'0': 'value'}
(X) {'0': 'value'} None
( ) SyntaxError: dynamic constant assignment error

Explanation:

https://docs.python.org/3/library/stdtypes.html#mapping-types-dict

>>> d, k, v = {}, '0', 'value'
>>> d[k] = v
>>> d
{'0': 'value'}

None will be printed as well b because spam(d, k, v) does not return anything.

Question >>>

What is the output of the following code?

```
def spam(x, sum):
    return sum if x == 0 else sum + spam(x-1, sum)
print(spam(3, 0))
```

( ) SyntaxError: invalid syntax
(X) 0
( ) 6
( ) RecursionError: maximum recursion depth exceeded in comparison

Explanation:

>>> Call to spam
..... **x = 3**
..... **sum = 0**
1 | def spam(x, sum):
2 | return sum if x == 0 else sum + spam(x-1, sum)
>>> Call to spam
..... **x = 2**
..... **sum = 0**
1 | def spam(x, sum):
2 | return sum if x == 0 else sum + spam(x-1, sum)
>>> Call to spam
..... **x = 1**

..... **sum = 0**
1 | def spam(x, sum):
2 | return sum if x == 0 else sum + spam(x-1, sum)
>>> Call to spam
..... **x = 0**
..... **sum = 0**
1 | def spam(x, sum):
2 | return sum if x == 0 else sum + spam(x-1, sum)
<<< Return value from spam: **0**
2 | return sum if x == 0 else sum + spam(x-1, sum)
<<< Return value from spam: **0**
2 | return sum if x == 0 else sum + spam(x-1, sum)
<<< Return value from spam: **0**
2 | return sum if x == 0 else sum + spam(x-1, sum)
<<< Return value from spam: **0**

Question >>>

What do you call **end** in the print function call below?

```
print(x, end='')
```

( ) named argument
( ) positional argument
(X) keyword argument
( ) arbitrary argument

Explanation:
keyword argument: an argument preceded by an identifier (e.g. name=) in a function call or passed as a value in a dictionary preceded by **. For example, 3 and 5 are both keyword arguments in the following calls to complex():
complex(real=3, imag=5)
complex(**{'real': 3, 'imag': 5})

Question >>>

How will you call the code below if you want to print 1, 2, 3?

```
def spam(a, b, c=3):
    print(a, b, c)
```

[X] spam(1,2)
[X] spam(1,2,3)
[X] spam(b=2,a=1)
[ ] spam(a=1,2,c=3)
[ ] spam(a=1,2,3)

Explanation:

>>> spam(1,2)
1 2 3
>>> spam(1,2,3)
1 2 3
>>> spam(b=2,a=1)
1 2 3
>>> spam(a=1,2,c=3)
SyntaxError: positional argument follows keyword argument
>>> spam(a=1,2,3)
SyntaxError: positional argument follows keyword argument

Question >>>

Which of the option(s) is valid based on the following code?

```
1 def spam(a, b, c=3):
2     print(a, b, c)
3 spam(1, 2, c=3,)
```

[ ] a, b are positional arguments
[X] 1, 2 are positional arguments
[X] 1 2 3 will be printed
[ ] 3 is a positional argument
[ ] Invalid Syntax on Line 3

Explanation:

positional argument: an argument that is not a keyword argument. Positional arguments can appear at the beginning of an argument list and/or be passed as elements of an iterable preceded by *. For example, 3 and 5 are both positional arguments in the following calls:
complex(3, 5)
complex(*(3, 5))

Question >>>

What is the output of the following code?

```
123
456
789
```

(X) No output
( ) 123456789
( ) SyntaxError: invalid syntax
( ) NameError: name 123 is not defined

Explanation:
No output. No error. 123, 456, 789 are valid integers that have not been assigned to a variable.

Question >>>

What is the output of the following code?

```
>>> 2 ** 3 ** 2
```

( ) SyntaxError: invalid syntax
( ) 12
( ) 64
( ) 128
( ) 256
(X) 512

Explanation:
>>> 2 ** 3 ** 2 #1 3**2=9
512
>>> 2 ** 9
512

Question >>>

What is the output of the following code?

```
>>> 5 ** 0 ** 0
```

( ) SyntaxError: invalid syntax
( ) 1
(X) 5
( ) 0

Explanation:
>>> 5 ** 0 ** 0 #1 0**0=1
5
>>> 5 ** 1
5
Question >>>

What is the output of the following code?

```
i = 10
while len(str(i)) > 5:
    i-=1
    print(i, end='')
else:
    i+=1
    print(i, end='')
```

( ) 98765
( ) 987656
(X) 11
( ) 11 .. 99999 will be printed

Explanation:
>>> i = 10 #1 i == 10
>>> while len(str(i)) > 5: #2 len(str(10)) == len('10') == 2 > 5 is false
... i-=1
... print(i, end='')
>>> else: #3 proceed next
... i+=1 #4 i = i=10 + 1 = 11
... print(i, end='') #5 print **11**

Question >>>

What is the output of the following code?

```
>>> i = 0
>>> while i != 0: i -= 1
... else: i += 1
>>> i
```

( ) SyntaxError: invalid syntax
( ) 0
(X) 1
( ) 2

Explanation:

```
>>> i = 0                #1 i == 0
>>> while i != 0: i -= 1  #2 while i=0 != 0 is false
...     else: i += 1      #3 i = i=0 + 1
>>> i                    #4 print 1
```

Question >>>

What is the output of the following code?

```
i = 30
while i > 0 :
    i -= 10
    print('loop', end=' ')
    if i <= 10:
        print('break', end=' ')
        break
else:
    print("else", end=' ')
```

(X) loop loop break
( ) loop loop loop break
( ) loop loop break else
( ) loop loop loop break else

Explanation:

```
1 | i = 30
2 | while i > 0 :
3 |     i -= 10
.............. i = 20
4 |     print('loop', end=' ')
loop
5 |     if i <= 10:
2 | while i > 0 :
```

3 | i -= 10
..... **i = 10**
4 | print('loop', end=' ')
loop
5 | if i <= 10:
6 | print('break', end=' ')
break
7 | break
Question >>>

Which option(s) below will output

Hello World!

given the following code?

```
<<< INSERT CODE HERE >>>
print('World')
```

( ) print('Hello')
( ) print('Hello', ' ')
( ) print('Hello', sep=' ')
(X) print('Hello', end=' ')

Explanation:
https://docs.python.org/3/library/functions.html#print
>>> print('Hello'); print('World')
Hello
World
>>> print('Hello', ' '); print('World')
Hello
World
>>> print('Hello', sep=' '); print('World')
Hello
World
>>> print('Hello', end=' '); print('World')
Hello World
Question >>>

Which option(s) will result in the output

0 1 2

Given the following code

```
c, b, a = 2, 1, 0
a, c = c, b
b = b - c
<<< INSERT CODE HERE >>>
print(a, b, c)
```

[X] a, b, c = b, c, a
[X] c, b, a = a, c, b
[ ] b, c, a = a, b, c
[ ] a, c, b = c, b, a

Explanation:

```
>>> c, b, a = 2, 1, 0
>>> a, c = c, b
>>> b = b - c
>>> a, b, c          # checkpoint value
(2, 0, 1)
>>> a, b, c = b, c, a  # correct
>>> a, b, c
(0, 1, 2)
>>> a, b, c = 2, 0, 1  # revert checkpoint
>>> c, b, a = a, c, b  # correct
>>> a, b, c
(0, 1, 2)
>>> a, b, c = 2, 0, 1  # revert checkpoint
>>> b, c, a = a, b, c  # incorrect
>>> a, b, c
(1, 2, 0)
>>> a, b, c = 2, 0, 1  # revert checkpoint
>>> a, c, b = c, b, a  # incorrect
>>> a, b, c
(1, 2, 0)
```

Question >>>

What is the output of the following code?

```
>>> def upcase(text): return text.upper()
>>> x = upcase
>>> f = [str.lower, x, str.capitalize]
>>> def dofunc(f):
...     message = f('Hello')
...     print(message)
>>> dofunc(x)
```

( ) SyntaxError: invalid syntax
( ) Hello
( ) hello
(X) HELLO
( ) AttributeError: type object 'str' has no attribute 'capitalize'

Explanation:

```
>>> def upcase(text): return text.upper()
>>> x = upcase      #1 x == upcase == text.upper
>>> f = [str.lower, x, str.capitalize] # ignore line no effect on result
>>> def dofunc(f):  #3 f == x == upcase == text.upper
...     message = f('Hello')  #4 f == x == upcase == text.upper(Hello')
...     print(message)       #5 print HELLO
>>> dofunc(x)       #2 calls dofunc pass x == upcase == text.upper
```

Question >>>

Which option prints **HELLO WORLD** given the following code?

```
>>> def dofunc(text, b):
...     def lowcase(): return text.lower()
...     def upcase(): return text.upper()
...     return upcase if b else lowcase
```

[ ] >>> dofunc('Hello World', True)
[X] >>> dofunc('Hello World', True)()
[X] >>> dofunc('Hello World', (True))()
[X] >>> dofunc('Hello World', (False,))()

Explanation:

```
>>> def dofunc(text, b):
...     def lowcase(): return text.lower()
```

```
...     def upcase(): return text.upper()
...     return upcase if b else lowcase
>>> dofunc('Hello World', True)
<function dofunc.<locals>.upcase at 0x0000029E6351A700>
>>> dofunc('Hello World', True)()
'HELLO WORLD'
>>> dofunc('Hello World', (True))()
'HELLO WORLD'
>>> dofunc('Hello World', (False,))()
'HELLO WORLD'
```

Question >>>