Progress (100%)

Item 1/10

Python is an example of:

- [x] a high-level programming language
- [ ] a natural language
- [ ] a machine language

Next →

Retake Test | Back to Summary

Progress (100%)

Item 2/10

A complete set of known commands is called:

- (•) an instruction list
- ( ) a low-level list
- ( ) a machine list

← Prev Next →

Retake Test Back to Summary

Progress (100%)

Item 3/10

What is a source code?

- Machine code executed by computers
- A program written in a high-level programming language
- Another name for source file

← Prev Next →

Retake Test Back to Summary

Progress (100%)

Item 4/10

What do you call a computer program which directly executes instructions written in a programming language?

- A translator
- An interpreter
- A compiler

← Prev | Next →

Retake Test | Back to Summary

Progress (100%)

Item 5/10

What Python version is covered in this course?

- (•) Python 3
- ( ) Python 1
- ( ) Python 2

← Prev Next →

Retake Test Back to Summary

Progress (100%)

Item 6/10

What is CPython?

- A compiled language used to perform high-level programming functions
- **The default implementation of the Python programming language**
- It is another name for Cython, a superset of the Python programming language

← Prev Next →

Retake Test Back to Summary

Progress (100%)

Item 7/10

What is IDLE?

- (●) It's an acronym that stands for Integrated Development and Learning Environment for Python
- ( ) It's an acronym that stands for Interactive Development and Learning Extension
- ( ) It's a Python version

← Prev Next →

Retake Test Back to Summary

Progress (100%)

Item 8/10

What do you call a tool that lets you launch your code step by step and inspect it at each moment of execution?

- An editor
- A debugger
- A console

← Prev Next →

Retake Test Back to Summary

Progress (100%)

Item 9/10

How did Python, the programming language, get its name?

- Guido van Rossum named it after the Pythonidae - a family of large, nonvenomous snakes
- **Guido van Rossum named it to honor Monty Python's Flying Circus, a BBC comedy series popular in the 1970s**
- Guido van Rossum named it to honor Python of Catana, a dramatic poet of the time of Alexander the Great

← Prev Next →

Progress (100%)

Item 10/10

Which one of the following is an example of a Python file extension?

- pi
- py
- p

← Prev

Retake Test | Back to Summary

Progress (100%)

Item 1/10

The **escape character** owes its name to the fact that it:

- cannot be caught due to its high speed
- escapes from source files into the computer memory
- changes the meaning of the character next to it

Next →

Retake Test | Back to Summary

Progress (100%)

Item 2/10

The meaning of the **positional parameter** is determined by its:

- name
- position
- appearance

← Prev | Next →

Retake Test | Back to Summary

Progress (100%)

Item 3/10

The most important difference between integer and floating-point numbers lies in the fact that:

- they cannot be used simultaneously
- integers cannot be literals, while floats can
- **they are stored differently in the computer memory**

← Prev Next →

Retake Test Back to Summary

Progress (100%)

Item 4/10

The `0x` prefix means that the number after it is denoted as:

- an octal
- a hexadecimal
- a decimal

← Prev Next →

Retake Test Back to Summary

Progress (100%)

Item 5/10

The `//` operator:

- does not exist
- performs integer division
- performs regular division

← Prev Next →

Retake Test Back to Summary

Item 6/10

The result of the following addition:

```
123 + 0.0
```

- cannot be evaluated
- is equal to `123`
- **is equal to `123.0`**

← Prev Next →

Retake Test Back to Summary

Progress (100%)

Item 7/10

Only one of the following statements is true - which one?

- addition precedes multiplication
- **multiplication precedes addition**
- neither statement can be evaluated

← Prev Next →

Retake Test Back to Summary

Progress (100%)

Item 8/10

Right-sided binding means that the following expression:

```
1 ** 2 ** 3
```

will be evaluated:

- (•) from right to left
- ( ) in random order
- ( ) from left to right

← Prev Next →

Retake Test Back to Summary

Progress (100%)

Item 9/10

A **keyword** is a word that:

- (●) cannot be used as a variable name
- ( ) is the most important word in the whole program
- ( ) is the key to the puzzle

← Prev Next →

Retake Test Back to Summary

Progress (100%)

Item 10/10

A value returned by the `input()` function is:

- [x] a string
- [ ] an integer
- [ ] a float

← Prev

Retake Test | Back to Summary

Progress (100%)

Item 1/20

What is the output of the following snippet if the user enters two lines containing `2` and `4` respectively?

```
x = input()
y = input()
print(x + y)
```

- 6
- 2
- **24**
- 4

Next →

Retake Test | Back to Summary

Item 2/20

What is the output of the following snippet?

```
x = 1
y = 2
z = x
x = y
y = z
print(x, y)
```

- [ ] 2 2
- [x] 2 1
- [ ] 1 1
- [ ] 1 2

← Prev Next →

Retake Test Back to Summary

Item 3/20

What is the output of the following snippet if the user enters two lines containing `2` and `4` respectively?

```
x = int(input())
y = int(input())
x = x / y
y = y / x
print(y)
```

- `2.0`
- the code will cause a runtime error
- **`8.0`**
- `4.0`

← Prev Next →

Retake Test Back to Summary

Item 4/20

What is the output of the following snippet if the user enters two lines containing `2` and `4` respectively?

```
x = int(input())
y = int(input())
print(x + y)
```

- (•) `6`
- ( ) `24`
- ( ) `2`
- ( ) `4`

← Prev Next →

Retake Test Back to Summary

Progress (100%)

Item 5/20

Only one of the following statements is false - which one?

- the right argument of the `%` operator cannot be zero
- multiplication precedes addition
- the result of the `/` operator is always an integer value
- the `**` operator uses right-sided binding

← Prev Next →

Item 6/20

The `\n` digraph forces the `print()` function to:

- duplicate the character next to the digraph
- (•) break the output line
- output exactly two characters: `\` and `n`
- stop its execution

← Prev Next →

Retake Test Back to Summary

Progress (100%)

Item 7/20

The `0o` prefix means that the number after it is denoted as:

- decimal
- binary
- octal
- hexadecimal

← Prev Next →

Retake Test Back to Summary

Item 8/20

What is the output of the following snippet if the user enters two lines containing `3` and `6` respectively?

```
x = input()
y = int(input())
print(x * y)
```

- 18
- 36
- 666
- **333333**

← Prev Next →

Retake Test Back to Summary

Item 9/20

What is the output of the following snippet if the user enters two lines containing `11` and `4` respectively?

```
x = int(input())
y = int(input())
x = x % y
x = x % y
y = y % x
print(y)
```

- (•) 1
- ( ) 2
- ( ) 3
- ( ) 4

← Prev Next →

Retake Test Back to Summary

Progress (100%)

Item 10/20

The value **twenty point twelve times ten raised to the power of eight** should be written as:

- (●) `20.12E8`
- ( ) `20E12.8`
- ( ) `20.12E8.0`
- ( ) `20.12*10^8`

← Prev Next →

Retake Test Back to Summary

Item 11/20

The `print()` function can output values of:

- not more than five arguments
- just one argument
- any number of arguments (excluding zero)
- **any number of arguments (including zero)**

← Prev Next →

Retake Test Back to Summary

Progress (100%)

Item 12/20

The result of the following division:

```
1 / 1
```

- is equal to `1`
- cannot be evaluated
- cannot be predicted
- **is equal to `1.0`**

← Prev Next →

Retake Test Back to Summary

Item 13/20

Left-sided binding determines that the result of the following expression:

```
1 // 2 * 3
```

is equal to:

- (•) `0`
- ( ) `0.0`
- ( ) `4.5`
- ( ) `0.16666666666666666`

← Prev Next →

Retake Test Back to Summary

Item 14/20

What is the output of the following snippet if the user enters two lines containing `2` and `4` respectively?

```
x = int(input())
y = int(input())
x = x // y
y = y // x
print(y)
```

- `2.0`
- `8.0`
- **the code will cause a runtime error**
- `4.0`

← Prev Next →

Retake Test Back to Summary

Progress (100%)

Item 15/20

One of the following variables' names is illegal - which one?

- (•) `True`
- ( ) `TRUE`
- ( ) `true`
- ( ) `tRUE`

← Prev Next →

Retake Test Back to Summary

Progress (100%)

Item 16/20

The meaning of the **keyword parameter** is determined by:

- (●) the argument's name specified along with its value
- ( ) its connection with existing variables
- ( ) its position within the argument list
- ( ) its value

← Prev Next →

Retake Test Back to Summary

Progress (100%)

Item 17/20

What is the output of the following snippet?

```
y = 2 + 3 * 5.
print(Y)
```

- 17
- 25.
- the snippet will cause an execution error
- 17.0

← Prev Next →

Retake Test Back to Summary

Progress (100%)

Item 18/20

What is the output of the following snippet?

```
x = 1 / 2 + 3 // 3 + 4 ** 2
print(x)
```

- 17
- 8.5
- 8
- **17.5**

← Prev Next →

Retake Test Back to Summary

Item 19/20

What is the output of the following snippet?

```
z = y = x = 1
print(x, y, z, sep='*')
```

- [ ] `x*y*z`
- [x] `1*1*1`
- [ ] `1 1 1`
- [ ] `x y z`

← Prev Next →

Retake Test Back to Summary

Progress (100%)

Item 20/20

The ** operator:

- does not exist
- performs exponentiation
- performs duplicated multiplication
- performs floating-point multiplication

Prev

Retake Test

Back to Summary

Progress (100%)

Item 1/10

An operator able to check whether two values are not equal is coded as:

- not ==
- != (selected)
- <>

Next →

Retake Test | Back to Summary

Progress (100%)

Item 2/10

How many stars will the following snippet send to the console?

```
i = 2
while i >= 0:
    print("*")
    i -= 2
```

- one
- three
- two

← Prev Next →

Retake Test Back to Summary

Progress (100%)

Item 3/10

How many hashes will the following snippet send to the console?

```
for i in range(-1, 1):
    print("#")
```

- three
- one
- two

← Prev Next →

Retake Test Back to Summary

Progress (100%)

Item 4/10

What value will be assigned to the `x` variable?

```
z = 10
y = 0
x = z > y or z == y
```

- (•) `True`
- ( ) `False`
- ( ) `1`

← Prev Next →

Retake Test Back to Summary

Progress (100%)

Item 5/10

What is the output of the following code?

```
lst = [3, 1, -1]
lst[-1] = lst[-2]
print(lst)
```

- [x] `[3, 1, 1]`
- [ ] `[1, 1, -1]`
- [ ] `[3, -1, 1]`

← Prev Next →

Retake Test Back to Summary

Item 6/10

The second assignment:

```
vals = [0, 1, 2]
vals[0], vals[1] = vals[1], vals[2]
```

- (●) doesn't change the list's length
- ( ) shortens the list
- ( ) extends the list

← Prev Next →

Retake Test Back to Summary

Item 7/10

Take a look at the snippet and choose one of the following statements which is true:

```
nums = []
vals = nums
vals.append(1)
```

- (●) `nums` and `vals` are of the same length
- ( ) `vals` is longer than `nums`
- ( ) `nums` is longer than `vals`

← Prev Next →

Retake Test Back to Summary

Progress (100%)

Item 8/10

Take a look at the snippet and choose one of the following statements which is true:

```
nums = []
vals = nums[:]
vals.append(1)
```

- `nums` is longer than `vals`
- `nums` and `vals` are of the same length
- **(selected)** `vals` is longer than `nums`

← Prev Next →

Retake Test Back to Summary

Progress (100%)

Item 9/10

How many elements does the `l` list contain?

```
l = [0 for i in range(1, 3)]
```

- (•) two
- ( ) one
- ( ) three

← Prev | Next →

Retake Test | Back to Summary

Progress (100%)

Item 10/10

What is the output of the following snippet?

```
lst = [0, 1, 2, 3]
x = 1
for elem in lst:
    x *= elem
print(x)
```

- 1
- 0
- 6

← Prev

Retake Test | Back to Summary

Progress (100%)

Item 1/20

What is the output of the following snippet?

```
a = 1
b = 0
c = a & b
d = a | b
e = a ^ b
print(c + d + e)
```

- (•) 2
- ( ) 0
- ( ) 3
- ( ) 1

Next →

Retake Test | Back to Summary

Progress (100%)

Item 2/20

What is the output of the following snippet?

```
lst = [1, 2, 3]
for v in range(len(lst)):
    lst.insert(1, lst[v])
print(lst)
```

- [ ] `[3, 2, 1, 1, 2, 3]`
- [x] `[1, 1, 1, 1, 2, 3]`
- [ ] `[1, 2, 3, 3, 2, 1]`
- [ ] `[1, 2, 3, 1, 2, 3]`

← Prev Next →

Retake Test Back to Summary

Progress (100%)

Item 3/20

The value eventually assigned to `x` is equal to:

```
x = 1
x = x == x
```

- 1
- False
- True
- 0

← Prev Next →

Retake Test Back to Summary

Progress (100%)

Item 4/20

What is the output of the following snippet?

```
t = [[3-i for i in range (3)] for j in ran
s = 0
for i in range(3):
    s += t[i][i]
print(s)
```

- 4
- 6
- 02
- 7

← Prev Next →

Retake Test Back to Summary

Progress (100%)

Item 5/20

The second assignment:

```
vals = [0, 1, 2]
vals[0], vals[2] = vals[2], vals[0]
```

- shortens the list
- doesn't change the list
- extends the list
- **reverses the list**

← Prev Next →

Retake Test Back to Summary

Progress (100%)

Item 6/20

What is the output of the following snippet?

```
lst = [3, 1, -2]
print(lst[lst[-1]])
```

- (•) 1
- ( ) 3
- ( ) -2
- ( ) -1

← Prev Next →

Retake Test Back to Summary

Item 7/20

What is the output of the following snippet?

```
lst1 = [1, 2, 3]
lst2 = []
for v in lst1:
    lst2.insert(0, v)
print(lst2)
```

- [1, 2, 3]
- [3, 3, 3]
- [3, 2, 1] ✓
- [1, 1, 1]

← Prev Next →

Retake Test Back to Summary

Progress (100%)

Item 8/20

What is the output of the following snippet?

```
lst = [[0, 1, 2, 3] for i in range(2)]
print(lst[2][0])
```

- 2
- the snippet will cause a runtime error
- 1
- 0

← Prev Next →

Retake Test Back to Summary

Item 9/20

Take a look at the snippet, and choose the true statement:

```
nums = [1, 2, 3]
vals = nums
del vals[1:2]
```

- (●) `nums` and `vals` are of the same length
- ( ) the snippet will cause a runtime error
- ( ) `vals` is longer than `nums`
- ( ) `nums` is longer than `vals`

← Prev Next →

Retake Test Back to Summary

Item 10/20

How many stars (`*`) will the following snippet send to the console?

```
i = 0
while i <= 5 :
    i += 1
    if i % 2 == 0:
        break
    print("*")
```

- (•) one
- ( ) three
- ( ) zero
- ( ) two

← Prev Next →

Retake Test Back to Summary

Progress (100%)

Item 11/20

How many elements does the `lst` list contain?

```
lst = [i for i in range(-1, 2)]
```

- one
- two
- four
- three

← Prev Next →

Retake Test Back to Summary

Progress (100%)

Item 12/20

How many hashes (`#`) will the following snippet send to the console?

```
var = 1
while var < 10:
    print("#")
    var = var << 1
```

- eight
- four
- one
- two

Prev Next

Retake Test Back to Summary

Progress (100%)

Item 13/20

What value will be assigned to the `x` variable?

```
z = 10
y = 0
x = y < z and z > y or y > z and z < y
```

- `False`
- `0`
- `True` ✓
- `1`

← Prev | Next →

Retake Test | Back to Summary

Progress (100%)

Item 14/20

What is the output of the following snippet?

```
lst = [1, 2, 3, 4]
print(lst[-3:-2])
```

- [x] `[2]`
- [ ] `[2, 3]`
- [ ] `[]`
- [ ] `[2, 3, 4]`

← Prev Next →

Retake Test Back to Summary

Item 15/20

How many hashes (`#`) will the following snippet send to the console?

```
for i in range(1):
    print("#")
else:
    print("#")
```

- zero
- three
- two
- one

← Prev Next →

Retake Test Back to Summary

Progress (100%)

Item 16/20

How many hashes (`#`) will the following snippet send to the console?

```
var = 0
while var < 6:
    var += 1
    if var % 2 == 0:
        continue
    print("#")
```

- one
- two
- three
- zero

← Prev Next →

Retake Test Back to Summary

Item 17/20

How many stars (`*`) will the following snippet send to the console?

```
i = 0
while i <= 3 :
    i += 2
    print("*")
```

- (•) two
- ( ) one
- ( ) three
- ( ) zero

← Prev Next →

Retake Test Back to Summary

Item 18/20

Which of the following sentences is true?

```
nums = [1, 2, 3]
vals = nums[-1:-2]
```

- `vals` is longer than `nums`
- **`nums` is longer than `vals`**
- the snippet will cause a runtime error
- `nums` and `vals` are of the same length

← Prev Next →

Retake Test Back to Summary

Progress (100%)

Item 19/20

An operator able to check whether two values are equal is coded as:

- [ ] =
- [ ] ===
- [ ] !=
- [x] ==

← Prev Next →

Retake Test Back to Summary

Item 20/20

After execution of the following snippet, the sum of all `vals` elements will be equal to:

```
vals = [0, 1, 2]
vals.insert(0, 1)
del vals[1]
```

- (•) 4
- ( ) 3
- ( ) 5
- ( ) 2

← Prev

Retake Test | Back to Summary

Progress (100%)

Item 1/10

A function definition starts with the keyword:

- fun
- function
- def

Next →

Retake Test | Back to Summary

Progress (100%)

Item 2/10

A function definition:

- [x] must be placed before the first invocation
- [ ] may be placed anywhere inside the code after the first invocation
- [ ] must not be placed among other code

← Prev Next →

Retake Test Back to Summary

Progress (100%)

Item 3/10

A function parameter is a kind of variable accessible:

- (•) only inside the function
- ( ) anywhere in the code
- ( ) only after the function definition's completion

← Prev Next →

Retake Test Back to Summary

Progress (100%)

Item 4/10

A way of argument passing in which the order of the arguments determines the initial parameters' values is referred to as:

- ordered
- positional
- sequential

← Prev Next →

Retake Test Back to Summary

Item 5/10

Which of the following statements is false?

- The `return` keyword forces the function's execution to terminate
- The `return` keyword may cause the function to return a value
- **The `return` keyword forces the function to restart its execution**

← Prev Next →

Retake Test Back to Summary

Progress (100%)

Item 6/10

The `None` keyword designates:

- (●) a `None` value
- ( ) an empty instruction
- ( ) a function which doesn't return a value

← Prev Next →

Retake Test Back to Summary

Progress (100%)

Item 7/10

A variable defined outside a function:

- may not be accessed in any way inside the function
- may be freely accessed inside the function
- **may be read, but not written (something more is needed to do so)**

← Prev Next →

Retake Test Back to Summary

Progress (100%)

Item 8/10

If a list is passed into a function as an argument, deleting any of its elements inside the function using the `del` instruction:

- will cause a runtime error
- will not affect the argument
- **will affect the argument**

← Prev Next →

Retake Test Back to Summary

Progress (100%)

Item 9/10

What is the output of the following snippet?

```
def fun(in=2, out=3):
    return in * out

print(fun(3))
```

- 6
- the snippet is erroneous
- 9

← Prev | Next →

Retake Test | Back to Summary

Progress (100%)

Item 10/10

What is the output of the following snippet?

```
tup = (1, ) + (1, )
tup = tup + tup
print(len(tup))
```

- (•) 4
- ( ) 2
- ( ) the snippet is erroneous

← Prev

Retake Test | Back to Summary

Progress (100%)

Item 1/20

The fact that tuples belong to sequence types means:

- they are actually lists
- they can be extended using the `.append()` method
- they can be modified using the `del` instruction
- **they can be indexed and sliced like lists**

Next →

Progress (100%)

Item 2/20

What is the output of the following snippet?

```
def fun(x):
    x += 1
    return x

x = 2
x = fun(x + 1)
print(x)
```

- 3
- 5
- 4
- the code is erroneous

← Prev Next →

Progress (100%)

Item 3/20

What is the output of the following snippet?

```
dct = { 'one':'two', 'three':'one', 'two':
v = dct['one']
for k in range(len(dct)):
    v = dct[v]
print(v)
```

- (•) `two`
- ( ) `three`
- ( ) `('one', 'two', 'three')`
- ( ) `one`

← Prev Next →

Progress (100%)

Item 4/20

Assuming that `tuple` is a correctly created tuple, the fact that tuples are immutable means that the following instruction:

```
tuple[1] = tuple[1] + tuple[0]
```

- (●) is illegal
- ( ) can be executed if and only if the tuple contains at least two elements
- ( ) is fully correct
- ( ) may be illegal if the tuple contains strings

← Prev Next →

Progress (100%)

Item 5/20

The following snippet:

```
def func(a, b):
    return a ** a

print(func(2))
```

- will output `4`
- **is erroneous**
- will output `2`
- will return `None`

← Prev Next →

Progress (100%)

Item 6/20

What is the output of the following snippet?

```
def fun(inp=2, out=3):
    return inp * out
print(fun(out=2))
```

- (•) 4
- ( ) 2
- ( ) 6
- ( ) the snippet is erroneous

← Prev Next →

Progress (100%)

Item 7/20

What is the output of the following snippet?

```
def f(x):
    if x == 0:
        return 0
    return x + f(x - 1)

print(f(3))
```

- 3
- 1
- the code is erroneous
- 6

← Prev Next →

Progress (100%)

Item 8/20

What is the output of the following snippet?

```
list =  ['Mary', 'had', 'a', 'little', 'la
def list(lst):
    del lst[3]
    lst[3] = 'ram'
print(list(list))
```

- (●) the snippet is erroneous
- ( ) `['Mary', 'had', 'a', 'little', 'lamb']`
- ( ) `['Mary', 'had', 'a', 'lamb']`
- ( ) `['Mary', 'had', 'a', 'ram']`

← Prev Next →

Progress (100%)

Item 9/20

The following snippet:

```
def func1(a):
    return a ** a
def func2(a):
    return func1(a) * func1(a)
print(func2(2))
```

- is erroneous
- will output `4`
- will output `2`
- **will output `16`**

← Prev Next →

Progress (100%)

Item 10/20

What is the output of the following snippet?

```
def fun(x):
    if x % 2 == 0:
        return 1
    else:
        return

print(fun(fun(2)) + 1)
```

- None
- 2
- 1
- **the code will cause a runtime error**

← Prev Next →

Item 11/20

A function defined in the following way:

```
def function(x=0):
    return x
```

- must be invoked without arguments
- may be invoked with any number of arguments (including zero)
- **may be invoked without any argument, or with just one**
- must be invoked with exactly one argument

← Prev Next →

Progress (100%)

Item 12/20

Which of the following lines properly starts a function using two parameters, both with zeroed default values?

- `fun fun(a, b=0):`
- `def fun(a=b=0):`
- `fun fun(a=0, b):`
- **(selected)** `def fun(a=0, b=0):`

← Prev Next →

Item 13/20

What code would you insert into the commented line to obtain the output that reads:

Expected output:

```
a
b
c
```

Code:

```
dct = { }
lst = ['a', 'b', 'c', 'd']
for i in range(len(lst) - 1):
    dct[lst[i]] = ( lst[i], )
for i in sorted(dct.keys()):
    k = dct[i]
    # insert your code
```

- `print(k["0"])`
- `print(k)`
- `print(k['0'])`
- (●) `print(k[0])`

← Prev Next →

Progress (100%)

Item 14/20

A built-in function is a function which:

- has been placed within your code by another programmer
- has to be imported before use
- comes with Python, and is an integral part of Python
- is hidden from programmers

← Prev Next →

Progress (100%)

Item 15/20

What is the output of the following snippet?

```
tup = (1, 2, 4, 8)
tup = tup[1:-1]
tup = tup[0]
print(tup)
```

- the snippet is erroneous
- (2)
- 2
- (2,)

← Prev Next →

Progress (100%)

Item 16/20

What is the output of the following snippet?

```
def any():
    print(var + 1, end='')
var = 1
any()
print(var)
```

- 11
- 12
- 21
- 22

← Prev Next →

Progress (100%)

Item 17/20

What is the output of the following snippet?

```
def fun(x):
    global y
    y = x * x
    return y

fun(2)
print(y)
```

- (•) 4
- ( ) None
- ( ) the code will cause a runtime error
- ( ) 2

← Prev Next →

Item 18/20

Which of the following statements is false?

- The `None` value can be assigned to variables
- The `None` value can be compared with variables
- The `None` value cannot be used as an argument of arithmetic operators
- (●) The `None` value may not be used outside functions

← Prev Next →

Progress (100%)

Item 19/20

What is the output of the following snippet?

```
def fun(x, y, z):
    return x + 2 * y + 3 * z

print(fun(0, z=1, y=3))
```

- the snippet is erroneous
- 3
- 9
- 0

← Prev Next →

Progress (100%)

Item 20/20

Which of the following lines properly starts a parameterless function definition?

- `def fun():`
- `fun function():`
- `function fun():`
- `def fun:`

Prev

Progress (100%)

Item 1/10

A namespace is:

- [x] a space in which names exist
- [ ] a name with a space
- [ ] a space with a name

Next →

Retake Test | Back to Summary

Item 2/10

If you want to import `pi` from `math`, which line will you use?

- `from math import pi`
- `from pi import math`
- `import pi from math`

← Prev | Next →

Retake Test | Back to Summary

Item 3/10

Which of the following is true?

- [x] packages can contain modules
- [ ] modules can contain packages
- [ ] modules can contain modules

← Prev Next →

Retake Test Back to Summary

Item 4/10

When an exception occurs, we say that it has been:

- [x] raised
- [ ] thrown
- [ ] dropped

Prev Next

Retake Test Back to Summary

Item 5/10

Which of the approachable `except:` branches is taken into consideration when an exception occurs?

- (•) the first matching branch
- ( ) the last matching branch
- ( ) any of the matching branches

← Prev Next →

Retake Test Back to Summary

Item 6/10

An assertion can be used to:

- make the programmer more assertive
- stop the program when some data have improper values
- import a module

Prev Next

Retake Test Back to Summary

Item 7/10

A code point is:

- a point used to write a code
- a code containing a point
- a number which makes up a character

← Prev Next →

Retake Test Back to Summary

Item 8/10

`isalnum()` checks if a string contains only letters and digits, and this is:

- (•) a method
- ( ) a module
- ( ) a function

← Prev Next →

Retake Test Back to Summary

Item 9/10

The expression:

```
'mike' > 'Mike'
```

is

- False
- **True**
- erroneous

← Prev Next →

Retake Test Back to Summary

Item 10/10

The `+=` operator, when applied to strings, performs:

- multiplication
- subtraction
- concatenation

Prev

Retake Test | Back to Summary

Item 1/20

When a module is imported, its contents:

- are executed as many times as they are imported
- are ignored
- may be executed (explicitly)
- **are executed once (implicitly)**

Next →

Back to Summary

Item 2/20

UNICODE is a standard:

- (●) like ASCII, but much more expansive
- ( ) honored by the whole universe
- ( ) used by coders from universities
- ( ) for coding floating-point numbers

← Prev Next →

Back to Summary

Item 3/20

The top-most Python exception is named:

- `PythonException`
- `TopException`
- (•) `BaseException`
- `Exception`

← Prev Next →

Back to Summary

Item 4/20

Knowing that a function named `fun()` resides in a module named `mod`, choose the proper way to import it:

- `from fun import mod`
- `import fun`
- `import fun from mod`
- (●) `from mod import fun`

← Prev Next →

Back to Summary

Item 5/20

A function returning a list of all entities available in a module is called:

- `content()`
- `entities()`
- (•) `dir()`
- `listmodule()`

← Prev Next →

Back to Summary

Item 6/20

The following code:

```
print(3 * 'abc' + 'xyz')
```

prints:

- abcabcxyzxyz
- xyzxyzxyzxyz
- abcabcabcxyz
- abcxyzxyzxyz

← Prev Next →

Back to Summary

Item 7/20

ASCII is:

- (●) short for *American Standard Code for Information Interchange*
- ( ) a standard Python module name
- ( ) a predefined Python variable name
- ( ) a character name

← Prev Next →

Back to Summary

Item 8/20

A predefined Python variable that stores the current module name is called:

- (●) `__name__`
- ( ) `__module__`
- ( ) `__mod__`
- ( ) `__modname__`

← Prev Next →

Back to Summary

Item 9/20

UTF-8 is:

- a Python version name
- a synonym for *byte*
- the 9th version of the UTF standard
- a form of encoding Unicode code points

← Prev Next →

Back to Summary

Item 10/20

The following code:

```
x = '\''
print(len(x))
```

prints:

- 20
- 3
- 2
- 1 ●

← Prev Next →

Back to Summary

Item 11/20

The following statement:

```
from a.b import c
```

causes the import of:

- entity `c` from module `a` from package `b`
- (●) entity `c` from module `b` from package `a`
- entity `b` from module `a` from package `c`
- entity `a` from module `b` from package `c`

← Prev Next →

Back to Summary

Item 12/20

Knowing that a function named `fun()` resides in a module named `mod`, and it has been imported using the following line:

```
import mod
```

Choose the way it can be invoked in your code:

- `fun()`
- `mod::fun()`
- `mod->fun()`
- (●) `mod.fun()`

← Prev Next →

Back to Summary

Item 13/20

The following code:

```
print(float("1, 3"))
```

- raises a `ValueError` exception
- prints `13`
- prints `1,3`
- prints `1.3`

← Prev Next →

Back to Summary

Item 14/20

The following code:

```
print(ord('c') - ord('a'))
```

prints:

- [x] 2
- [ ] 0
- [ ] 3
- [ ] 1

← Prev Next →

Back to Summary

Item 15/20

The following code:

```
print('Mike' > "Mikey")
```

prints:

- (●) False
- ( ) 1
- ( ) True
- ( ) 0

← Prev Next →

Back to Summary

Progress (100%)

Item 16/20

The unnamed `except:` block:

- can be placed anywhere
- must be the first one
- cannot be used if any named block has been used
- **must be the last one**

← Prev
Next →

Back to Summary

Item 17/20

Entering the `try:` block implies that:

- the block will be omitted
- some of the instructions from this block may not be executed
- all of the instructions from this block will be executed
- none of the instructions from this block will be executed

← Prev Next →

Back to Summary

Item 18/20

The `pyc` file contains:

- (●) compiled Python code
- ( ) a Python compiler
- ( ) Python source code
- ( ) a Python interpreter

← Prev Next →

Back to Summary

Item 19/20

The following code:

```
print(chr(ord('z') - 2))
```

prints:

- (●) x
- ( ) z
- ( ) y
- ( ) a

← Prev Next →

Back to Summary

Item 20/20

The following statement:

```
assert var == 0
```

- will stop the program when `var == 0`
- is erroneous
- (●) will stop the program when `var != 0`
- has no effect

Prev

Back to Summary

Progress (100%)

Item 1/10

A subclass is usually:

- a twin of its superclass
- more specialized than its superclass
- more general than its superclass

Next →

Retake Test | Back to Summary

Progress (100%)

Item 2/10

An object is characterized by the following three:

- properties, estates, lands
- **name, properties, activities**
- name, owner, possession

Prev Next

Retake Test Back to Summary

Progress (100%)

Item 3/10

An alternative name for a data structure called a *stack* is:

- FIFO
- LIFO
- FOLO

Prev Next

Retake Test Back to Summary

Item 4/10

A variable existing as a separate being in separate objects is called:

- a separate variable
- an instance variable
- an objective variable

← Prev Next →

Retake Test Back to Summary

Item 5/10

A function able to check if an object is equipped with a given property is named:

- `hasprop()`
- **`hasattr()`**
- `hasvar()`

← Prev Next →

Retake Test Back to Summary

Item 6/10

Is there a way to check if a class is a subclass of another class?

- (•) yes, there is a function able to do that
- ( ) no
- ( ) it may be possible, but only under special conditions

← Prev Next →

Retake Test Back to Summary

Item 7/10

The function named `super()` may be used to:

- make a class better
- make a class super
- **access a super class's attributes and/or methods**

← Prev Next →

Retake Test Back to Summary

Item 8/10

A user defined exception:

- may be derived from the Exception class
- must not be derived from the Exception class
- must be derived from the Exception class

← Prev Next →

Retake Test Back to Summary

Item 9/10

The two basic, mutually exclusive, file open modes are named:

- [x] binary and text
- [ ] text and image
- [ ] binary and ternary

Prev Next

Retake Test Back to Summary

Item 10/10

A method able to read data from a file into a byte array object, is named:

- `readin()`
- `readout()`
- `readinto()`

Prev

Retake Test | Back to Summary

Item 1/20

What will be the result of executing the following code?

```
try:
    raise Exception(1,2,3)
except Exception as e:
    print(len(e.args))
```

- (●) it will print 3
- ( ) it will print 1
- ( ) it will raise an unhandled exception
- ( ) it will print 2

Next →

Retake Test | Back to Summary

Item 2/20

What will be the output of the following code?

```
class A:
    X = 0
    def __init__(self,v = 0):
        self.Y = v
        A.X += v

a = A()
b = A(1)
c = A(2)
print(c.X)
```

- 1
- 3
- 2
- 0

← Prev Next →

Retake Test Back to Summary

Item 3/20

What will be the result of executing the following code?

```
class A:
    v = 2

class B(A):
    v = 1


class C(B):
    pass

o = C()
print(o.v)
```

- (●) it will print 1
- ( ) it will print 2
- ( ) it will print an empty line
- ( ) it will raise an exception

← Prev Next →

Retake Test Back to Summary

Item 4/20

What will be the result of executing the following code?

```
def o(p):
    def q():
        return '*' * p
    return q

r = o(1)
s = o(2)
print(r() + s())
```

- it will print `*`
- **it will print `***`**
- it will print `**`
- it will print `****`

← Prev Next →

Retake Test Back to Summary

Item 5/20

If there is a superclass named `A` and a subclass named `B`, which one of the presented invocations should you put instead of the comment?

```
class A:
    def __init__(self):
        self.a = 1

class B(A):
    def __init__(self):
        # Put selected line here.
        self.b = 2
```

- `A.__init__()`
- `__init__()`
- `A.__init__(self)` ●
- `A.__init__(1)`

← Prev Next →

Retake Test Back to Summary

Item 6/20

What will be the effect of running the following code?

```
class A:
    def __init__(self,v):
        self.__a = v + 1

a = A(0)
print(a.__a)
```

1

2

it will raise an `AttributeError` exception

0

← Prev Next →

Retake Test Back to Summary

Item 7/20

What will be the result of executing the following code?

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

a = A(1)
print(hasattr(a,'A'))
```

- (●) it will raise an exception
- ( ) `True`
- ( ) `False`
- ( ) `1`

← Prev Next →

Retake Test Back to Summary

Item 8/20

What will be the result of executing the following code?

```
def I():
    s = 'abcdef'
    for c in s[::2]:
        yield c
for x in I():
    print(x,end='')
```

- (●) it will print `ace`
- ( ) it will print `bdf`
- ( ) it will print `abcdef`
- ( ) it will print an empty line

← Prev Next →

Retake Test Back to Summary

Item 9/20

What will be the result of executing the following code?

```
class A:
    def __str__(self):
        return 'a'

class B:
    def __str__(self):
        return 'b'


class C(A, B):
    pass

o = C()
print(o)
```

- it will print `b`
- it will raise an exception
- it will print `a`
- it will print `c`

← Prev Next →

Retake Test Back to Summary

Item 10/20

A data structure described as *LIFO* is actually a:

- (●) stack
- ( ) tree
- ( ) heap
- ( ) list

← Prev Next →

Retake Test Back to Summary

Item 11/20

What will be the output of the following code?

```
class A:
    def __init__(self,v = 1):
        self.v = v

    def set(self,v):
        self.v = v
        return v

a = A()
print(a.set(a.v + 1))
```

1

0

2

3

← Prev

Next →

Retake Test

Back to Summary

Item 12/20

What will be the output of the following code?

```
class A:
    A = 1

print(hasattr(A,'A'))
```

- 0
- 1
- True
- False

← Prev Next →

Retake Test Back to Summary

Item 13/20

What will be the result of executing the following code?

```
class I:
    def __init__(self):
        self.s = 'abc'
        self.i = 0
    def __iter__(self):
        return self
    def __next__(self):
        if self.i == len(self.s):
                raise StopIteration
        v = self.s[self.i]
        self.i += 1
        return v
for x in I():
    print(x,end='')
```

- it will print `cba`
- **it will print `abc`**
- it will print `012`
- it will print `210`

← Prev Next →

Retake Test Back to Summary

Item 14/20

What will be the result of executing the following code?

```
class A:
    def a(self):
        print('a')

class B:
    def a(self):
        print('b')


class C(B,A):
    def c(self):
        self.a()

o = C()
o.c()
```

- it will raise an exception
- it will print `a`
- **it will print `b`**
- it will print `c`

← Prev Next →

Retake Test Back to Summary

Item 15/20

What will be the result of executing the following code?

```
class A:
    pass

class B(A):
    pass

class C(B):
    pass

print(issubclass(C,A))
```

- it will print `1`
- it will print `False`
- (•) it will print `True`
- it will raise an exception

← Prev Next →

Retake Test Back to Summary

Item 16/20

What will be the result of executing the following code?

```
def fun(n):
    s = '+'
    for i in range(n):
        s += s
        yield s

for x in fun(2):
    print(x, end='')
```

- it will print `++`
- it will print `+`
- it will print `+++`
- **it will print `++++++`**

← Prev Next →

Retake Test Back to Summary

Item 17/20

What will be the result of executing the following code?

```
class Ex(Exception):
    def __init__(self,msg):
        Exception.__init__(self,msg + msg)
        self.args = (msg,)

try:
    raise Ex('ex')
except Ex as e:
    print(e)
except Exception as e:
    print(e)
```

- it will raise an unhandled exception
- it will print `exex`
- it will print an empty line
- **it will print `ex`**

← Prev Next →

Retake Test Back to Summary

Item 18/20

What will be the result of executing the following code?

```
class A:
    def __str__(self):
        return 'a'

class B(A):
    def __str__(self):
        return 'b'


class C(B):
    pass

o = C()
print(o)
```

- it will print `a`
- (•) it will print `b`
- it will raise an exception
- it will print `c`

← Prev Next →

Retake Test Back to Summary

Item 19/20

If the class's constructor is declared as below, which one of the assignments is valid?

```
class Class:
    def __init__(self):
        pass
```

- `object = Class(self)`
- `object = Class(object)`
- `object = Class`
- ● `object = Class()`

← Prev Next →

Retake Test Back to Summary

Item 20/20

What will be the result of executing the following code?

```
def f(x):
    try:
        x = x / x
    except:
        print("a",end='')
    else:
        print("b",end='')
    finally:
        print("c",end='')

f(1)
f(0)
```

- it will print `acac`
- (●) it will print `bcac`
- it will raise an unhandled exception
- it will print `bcbc`

Prev

Retake Test

Back to Summary

Progress (100%)

Item 1/30

Which of the following sentences is true?

```
nums = [1, 2, 3]
vals = nums
```

- `vals` is longer than `nums`
- `nums` is longer than `vals`
- `nums` and `vals` are different lists
- (●) `nums` and `vals` are different names of the same list

Next →

Retake Test | Back to Summary

Progress (100%)

Item 2/30

What is the output of the following snippet?

```
def fun(x):
    if x % 2 == 0:
        return 1
    else:
        return 2

print(fun(fun(2)))
```

- the code will cause a runtime error
- `1`
- `2` (selected)
- `2None`

← Prev
Next →

Retake Test
Back to Summary

Progress (100%)

Item 3/30

What is the output of the following piece of code?

```
x = 1
y = 2
x, y, z = x, x, y
z, y, z = x, y, z

print(x, y, z)
```

- 1 2 2
- **1 1 2**
- 1 2 1
- 2 1 2

← Prev
Next →

Retake Test
Back to Summary

Progress (100%)

Item 4/30

Take a look at the snippet and choose the true statement:

```
nums = [1, 2, 3]
vals = nums
del vals[:]
```

- `vals` is longer than `nums`
- `nums` and `vals` are different lists
- **(selected)** `nums` and `vals` are different names of the same list
- the snippet will cause a runtime error
- `nums` is longer than `vals`

← Prev
Next →

Progress (100%)

Item 5/30

The result of the following division:

```
1 // 2
```

- is equal to `0.0`
- **is equal to `0`**
- is equal to `0.5`
- cannot be predicted

← Prev
Next →

Retake Test
Back to Summary

Progress (100%)

Item 6/30

What is the output of the following snippet?

```
dct = { 'one':'two', 'three':'one', 'two':
v = dct['three']

for k in range(len(dct)):
    v = dct[v]

print(v)
```

- ('one', 'two', 'three')
- **one**
- two
- three

← Prev Next →

Retake Test Back to Summary

Progress (100%)

Item 7/30

One of the following variable names is illegal - which one?

- `in_`
- `in`
- `IN`
- `In`

← Prev | Next →

Retake Test | Back to Summary

Progress (100%)

Item 8/30

What is the output of the following snippet?

```
def fun(x, y):
    if x == y:
        return x
    else:
        return fun(x, y-1)

print(fun(0, 3))
```

- 2
- 1
- 0
- the snippet will cause a runtime error

← Prev
Next →

Retake Test
Back to Summary

Progress (100%)

Item 9/30

What is the output of the following snippet?

```
def fun(inp=2, out=3):
    return inp * out
print(fun(out=2))
```

- 2
- the snippet is erroneous
- 4
- 6

← Prev
Next →

Retake Test
Back to Summary

Progress (100%)

Item 10/30

Assuming that the `tuple` is a correctly created tuple, the fact that tuples are immutable means that the following instruction:

```
tuple[1] = tuple[1] + tuple[0]
```

- (●) is illegal
- ( ) may be illegal if the tuple contains strings
- ( ) is fully correct
- ( ) can be executed if and only if the tuple contains at least two elements

← Prev
Next →

Retake Test
Back to Summary

Progress (100%)

Item 11/30

What is the output of the following snippet?

```
list =  [x * x for x in range(5)]
def fun(lst):
    del lst[lst[2]]
    return lst

print(fun(list))
```

- [ ] `[0, 1, 4, 16]`
- [ ] `[0, 1, 9, 16]`
- [x] `[0, 1, 4, 9]`
- [ ] `[1, 4, 9, 16]`

← Prev
Next →

Retake Test
Back to Summary

Progress (100%)

Item 12/30

What is the output of the following piece of code?

```
x = 1 // 5 + 1 / 5
print(x)
```

- 0
- 0.0
- 0.2
- 0.4

← Prev
Next →

Retake Test
Back to Summary

Progress (100%)

Item 13/30

How many elements does the `lst` list contain?

```
lst = [i for i in range(-1, -2)]
```

- three
- one
- zero
- two

← Prev | Next →

Retake Test | Back to Summary

Progress (100%)

Item 14/30

What is the output of the following snippet?

```
dct = {}
dct['1'] = (1, 2)
dct['2'] = (2, 1)

for x in dct.keys():
    print(dct[x][1],end="")
```

- (2,1)
- (1,2)
- 12
- 21

← Prev
Next →

Retake Test
Back to Summary

Progress (100%)

Item 15/30

What is the output of the following piece of code if the user enters two lines containing `2` and `4` respectively?

```
x = float(input())
y = float(input())
print(y ** (1 / x))
```

- 0.0
- 4.0
- **2.0**
- 1.0

← Prev
Next →

Retake Test
Back to Summary

Progress (100%)

Item 16/30

How many hashes (`#`) will the following snippet send to the console?

```
lst = [[x for x in range(3)] for y in rang

for r in range(3):
    for c in range(3):
        if lst[r][c] % 2 != 0:
            print("#")
```

- six
- nine
- three
- zero

← Prev
Next →

Retake Test
Back to Summary

Progress (100%)

Item 17/30

What is the output of the following piece of code if the user enters two lines containing `3` and `6` respectively?

```
y = input()
x = input()
print(x + y)
```

- (•) 63
- ( ) 36
- ( ) 3
- ( ) 6

← Prev
Next →

Retake Test
Back to Summary

Progress (100%)

Item 18/30

The following snippet:

```
def func(a, b):
    return b ** a

print(func(b=2, 2))
```

- will output `None`
- is erroneous
- will output `4`
- will output `2`

← Prev
Next →

Retake Test
Back to Summary

Progress (100%)

Item 19/30

What is the output of the following piece of code?

```
print("a", "b", "c", sep="sep")
```

- abc
- a b c
- asepbsepcsep
- **asepbsepc**

← Prev
Next →

Retake Test
Back to Summary

Progress (100%)

Item 20/30

What value will be assigned to the `x` variable?

```
z = 0
y = 10
x = y < z and z > y or y > z and z < y
```

- False
- 0
- **True**
- 1

← Prev | Next →

Retake Test | Back to Summary

Progress (100%)

Item 21/30

The meaning of a *positional argument* is determined by:

- its value
- its position within the argument list
- its connection with existing variables
- the argument's name specified along with its value

← Prev
Next →

Retake Test
Back to Summary

Progress (100%)

Item 22/30

What is the output of the following snippet?

```
tup = (1, 2, 4, 8)
tup = tup[-2:-1]
tup = tup[-1]
print(tup)
```

- (•) `4`
- ( ) `(4,)`
- ( ) `(4)`
- ( ) `44`

← Prev
Next →

Retake Test
Back to Summary

Progress (100%)

Item 23/30

The following snippet:

```
def func1(a):
    return None

def func2(a):
    return func1(a) * func1(a)

print(func2(2))
```

- will output `2`
- will output `16`
- **will cause a runtime error**
- will output `4`

← Prev
Next →

Retake Test
Back to Summary

Progress (100%)

Item 24/30

What is the output of the following snippet?

```
dd = { "1":"0", "0":"1" }
for x in dd.vals():
    print(x, end="")
```

- `1 0`
- `0 1`
- the code is erroneous (the `dict` object has no `vals()` method)
- `0 0`

← Prev
Next →

Retake Test
Back to Summary

Progress (100%)

Item 25/30

How many stars (`*`) will the following snippet send to the console?

```
i = 0
while i < i + 2 :
    i += 1
    print("*")
else:
    print("*")
```

- zero
- two
- one
- **the snippet will enter an infinite loop, printing one star per line**

← Prev
Next →

Retake Test
Back to Summary

Progress (100%)

Item 26/30

What will be the output of the following snippet?

```
a = 1
b = 0
a = a ^ b
b = a ^ b
a = a ^ b

print(a, b)
```

- 1 0
- **0 1**
- 0 0
- 1 1

← Prev
Next →

Retake Test
Back to Summary

Progress (100%)

Item 27/30

An operator able to check whether two values are not equal is coded as:

- `not ==`
- `!=`
- `<>`
- `=/=`

← Prev Next →

Retake Test Back to Summary

Progress (100%)

Item 28/30

What is the output of the following piece of code if the user enters two lines containing `3` and `2` respectively?

```
x = int(input())
y = int(input())
x = x % y
x = x % y
y = y % x
print(y)
```

- (•) 0
- ( ) 3
- ( ) 2
- ( ) 1

← Prev
Next →

Retake Test
Back to Summary

Progress (100%)

Item 29/30

What is the output of the following snippet?

```
lst = [1, 2]

for v in range(2):
    lst.insert(-1, lst[v])

print(lst)
```

- [ ] `[1, 2, 1, 2]`
- [x] `[1, 1, 1, 2]`
- [ ] `[2, 1, 1, 2]`
- [ ] `[1, 2, 2, 2]`

← Prev
Next →

Retake Test
Back to Summary

Progress (100%)

Item 30/30

Which of the following lines incorrectly invokes the function defined as:

```
def fun(a, b, c=0):
```

- `fun(0, 1, 2)`
- `fun(b=1)`
- `fun(a=0, b=0)`
- `fun(a=1, b=0, c=0)`

← Prev

Retake Test
Back to Summary

Progress (100%)

Item 1/30

The following code:

```
print(float("1.3"))
```

- (●) prints `1.3`
- ( ) prints `13`
- ( ) prints `1,3`
- ( ) raises a `ValueError` exception

Next →

Retake Test | Back to Summary

Item 2/30

The following statement:

```
assert var != 0
```

- has no effect
- will stop the program when `var == 0`
- is erroneous
- **will stop the program when `var != 0`**

Prev Next

Retake Test Back to Summary

Item 3/30

What will be the output of the following snippet?

```
try:
    raise Exception
except:
    print("c")
except BaseException:
    print("a")
except Exception:
    print("b")
```

- it will cause an error
- `b`
- `a`
- `1`
- `1`

Prev Next

Retake Test Back to Summary

Item 4/30

What will be the result of executing the following code?

```
try:
    raise Exception(1, 2, 3)
except Exception as e:
    print(len(e.args))
```

- it will print 2
- **it will print 3**
- it will raise an unhandled exception
- it will print 1

Prev Next

Retake Test Back to Summary

Item 5/30

If `s` is a stream opened in *read* mode, the following line:

```
q = s.read(1)
```

will read:

- (•) one character from the stream
- ( ) one buffer from the stream
- ( ) one line from the stream
- ( ) one kilobyte from the stream

Prev Next

Retake Test Back to Summary

Item 6/30

What will be the result of executing the following code?

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

a = A(1)
print(hasattr(a, 'A'))
```

- it will print `True`
- it will print `False`
- **it will raise an exception**
- it will print `1`
- it will print `1`

← Prev Next →

Retake Test Back to Summary

Item 7/30

What will be the result of executing the following code?

```
def o(p):
    def q():
        return '*' * p
    return q

r = o(1)
s = o(2)
print(r() + s())
```

- it will print `*`
- **it will print `***`**
- it will print `**`
- it will print `****`
- it will print `****`

← Prev Next →

Retake Test Back to Summary

Item 8/30

The following statement:

```
from a.b import c
```

causes the import of:

- (●) entity `c` from module `b` from package `a`
- ( ) entity `c` from module `a` from package `b`
- ( ) entity `b` from module `a` from package `c`
- ( ) entity `a` from module `b` from package `c`

← Prev Next →

Retake Test Back to Summary

Item 9/30

What will be the output of the following snippet?

```
try:
    raise Exception
except BaseException:
    print("a")
except Exception:
    print("b")
except:
    print("c")
```

- a
- b
- 1
- it will cause an error

Prev Next

Retake Test Back to Summary

Item 10/30

The `sys.stderr` stream is normally associated with:

- a null device
- the screen
- the printer
- the keyboard

Prev Next

Retake Test Back to Summary

Item 11/30

What will be the result of executing the following code?

```
class A:
    pass

class B(A):
    pass

class C(B):
    pass

print(issubclass(A, C))
```

- it will raise an exception
- it will print `1`
- it will print `False`
- it will print `True`
- it will print `True`

← Prev Next →

Retake Test Back to Summary

Item 12/30

What will be the result of executing the following code?

```
def I(n):
    s = '+'
    for i in range(n):
        s += s
        yield s

for x in I(2):
    print(x, end='')
```

- it will print `++++++`
- it will print `+`
- it will print `+++`
- it will print `++`

Prev Next

Retake Test Back to Summary

Item 13/30

If you want to fill a byte array with data read in from a stream, you can use:

- the `readbytes()` method
- the `readfrom()` method
- the `readinto()` method
- the `read()` method
- the `read()` method

Prev Next

Retake Test Back to Summary

Item 14/30

What will be the result of executing the following code?

```
class A:
    def a(self):
        print('a')

class B:
    def a(self):
        print('b')


class C(B, A):
    def c(self):
        self.a()

o = C()
o.c()
```

- it will raise an exception
- it will print `a`
- it will print `c`
- it will print `b`

Prev Next

Retake Test Back to Summary

Item 15/30

What will be the result of executing the following code?

```
class I:
    def __init__(self):
        self.s = 'abc'
        self.i = 0
    def __iter__(self):
        return self
    def __next__(self):
        if self.i == len(self.s):
                raise StopIteration
        v = self.s[self.i]
        self.i += 1
        return v

for x in I():
    print(x, end='')
```

- it will print `210`
- it will print `cba`
- **it will print `abc`**
- it will print `012`
- it will print `012`

Prev Next

Retake Test Back to Summary

Item 16/30

What will be the output of the following code, located in the `p.py` file?

```
print(__name__)
```

- `__p.py__`
- `p.py`
- `main`
- `__main__` ✓

← Prev Next →

Retake Test Back to Summary

Item 17/30

The following code:

```
x = "\\\\"
print(len(x))
```

- prints 3
- will cause an error
- prints 1
- prints 2

← Prev Next →

Retake Test Back to Summary

Item 18/30

What will be the effect of running the following code?

```
class A:
    def __init__(self, v):
        self.__a = v + 1

a = A(0)
print(a.__a)
```

- it will raise an `AttributeError` exception
- it will print `2`
- it will print `1`
- it will print `0`

← Prev Next →

Retake Test Back to Summary

Item 19/30

Knowing that a function named `fun()` resides in a module named `mod`, and was imported using the following statement:

```
from mod import fun
```

Choose the right way to invoke the `fun()` function:

- `mod:fun()`
- **`fun()`**
- `mod.fun()`
- `mod::fun()`
- `mod::fun()`

← Prev Next →

Retake Test Back to Summary

Item 20/30

The following line of code:

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

- may be valid if `line` is a list
- **is valid as `open` returns an iterable object**
- is invalid as `open` returns nothing
- is invalid as `open` returns a non-iterable object

← Prev Next →

Retake Test Back to Summary

Item 21/30

What will be the output of the following code?

```
class A:
    A = 1
    def __init__(self):
        self.a = 0

print(hasattr(A, 'a'))
```

- 1
- **False**
- 0
- True

← Prev Next →

Retake Test Back to Summary

Item 22/30

The following code prints:

```
print(chr(ord('p') + 2))
```

- t
- q
- r
- s
- 3

Prev Next

Retake Test Back to Summary

Item 23/30

The following code:

```
x = "\\\"
print(len(x))
```

- prints 1
- prints 2
- prints 3
- will cause an error

Prev Next

Retake Test Back to Summary

Item 24/30

If the class's constructor is declared as below, which one of the assignments is invalid?

```
class Class:
    def __init__(self, val=0):
        pass
```

- `object = Class(None)`
- **`object = Class(1, 2)`**
- `object = Class(1)`
- `object = Class()`

`object = Class()`

← Prev Next →

Retake Test Back to Summary

Item 25/30

What output will appear after running the following snippet?

```
import math
print(dir(math))
```

- the number of all the entities residing in the `math` module
- **a list of all the entities residing in the `math` module**
- an error message
- a string containing the fully qualified name of the module

← Prev Next →

Retake Test Back to Summary

Item 26/30

Assuming that the following three files: `a.py`, `b.py`, and `c.py` reside in the same folder, what will be the output produced after running the `c.py` file?

```
# file a.py
print("a", end='')

#file b.py
import a
print("b", end='')

#file c.py
print("c", end='')
import a
import b
```

- `abc`
- `bac`
- `cba`
- `cab`

← Prev Next →

Retake Test Back to Summary

Item 27/30

What will be the output of the following code?

```
class A:
    def __init__(self, v=2):
        self.v = v

    def set(self, v=1):
        self.v += v
        return self.v

a = A()
b = a
b.set()
print(a.v)
```

- 3
- 1
- 0
- 2

Prev Next

Retake Test Back to Summary

Progress (100%)

Item 28/30

The compiled Python bytecode is stored in files which have their names ending with:

- `py`
- `pc`
- `pyc`
- `pyb`

Prev Next

Retake Test Back to Summary

Item 29/30

Assuming that the `open()` invocation has gone successfully, the following snippet will:

```
for x in open('file', 'rt'):
    print(x)
```

- read the file character by character
- read the whole file at once
- **read the file line by line**
- cause an exception

Prev Next

Retake Test Back to Summary

Item 30/30

If there are more than one `except:` branches after the `try:` clause, we can say that:

- one or more of the `except:` blocks will be executed
- exactly one of the `except:` blocks will be executed
- none of the `except:` blocks will be executed
- **not more than one `except:` block will be executed**

Prev

Retake Test

Back to Summary