Code of Honor

I pledge to follow the Code of Honor and obey the rules for taking this test:

- I will work entirely alone on this test, and all the solutions I submit will be my own work;
- I will not share my solutions to the test with anyone;
- I will not use a false identity, or take this test in somebody else's name;
- I will not engage in any unfair activities that will dishonestly improve my results or influence somebody else's results.

Python Essentials - Part 1 Summary Test

Time limit: 45 minutes
Number of questions: 30
Points to score: 30
Passing score: 70%

Start

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)
```

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

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="")
```

- [ ] `(1,2)`
- [ ] `12`
- [ ] `(2,1)`
- [x] `21`

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**
- 2 1 2
- 1 2 1

What is the output of the following piece of code?

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

- 0.4
- **0.2**
- 0.0
- 0

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

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

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

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

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)
```

- `3`
- `6`
- `36` (selected)
- `63`

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

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

- [x] `nums` and `vals` are different names of the same list
- [ ] `nums` is longer than `vals`
- [ ] `nums` and `vals` are different lists
- [ ] `vals` is longer than `nums`
- [ ] the snippet will cause a runtime error

What is the output of the following snippet?

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

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

print(v)
```

- [ ] `three`
- [ ] `('one', 'two', 'three')`
- [x] `one`
- [ ] `two`

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

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

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

- [x] three
- [ ] zero
- [ ] six
- [ ] nine

The following snippet:

```
def func1(a):
    return None

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

print(func2(2))
```

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

Which of the following sentences is true?

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

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

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]
```

- [x] 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

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

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

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

How many elements does the `lst` list contain?

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

- [x] `zero`
- [ ] `two`
- [ ] `one`
- [ ] `three`

The result of the following division:

```
1 // 2
```

- is equal to `0.5`
- is equal to `0.0`
- (•) is equal to `0`
- cannot be predicted

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

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

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))
```

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

What is the output of the following piece of code?

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

- asepbsepc
- a b c
- abc
- asepbsepcsep

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
```

- 1
- True
- 0
- False

The following snippet:

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

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

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

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))
```

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

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

- `IN`
- `in_`
- `in` ✓
- `In`

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

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

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

What is the output of the following snippet?

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

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

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

What is the output of the following snippet?

```
lst = [1, 2]

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

print(lst)
```

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

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`
- `1 1`
- `0 0`

What is the output of the following snippet?

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

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

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