Exam 1

Consider the following code snippet. Which of the following options will **not** be an **int** data type?

```
q = 100//10
```

- q + 5
- q/1.0 ✔
- q
- q ** 3
- q*3

Which of the following is an **invalid** variable name?

- found_prime
- count_2020
- expenditure$1000 ✔
- ___terminate
- exam1

What kind of error is present in the following line of code?

```
for = 10
```

- Runtime error
- Syntax error ✔
- Indentation error
- There isn't any error

Select the statement(s) that are executed **exactly once** during a function invocation.

- [ ] continue
- [ ] while
- [ ] for
- [ ] break
- [ ] return ✔

What values of i and j will print "Data" as output from the following code snippet?

```
i = ???
j = ???

text = "In Data we trust"
print(text[i:j])
```

- ```
  i = 4
  j = 8
  ```
- ```
  i = -14
  j = -10
  ```
- ```
  i = 3
  j = 7
  ```
  ✔
- ```
  i = -1
  j = -5
  ```
- ```
  i = 2
  j = 6
  ```

Hint for the question below: Open Python interactive mode and type the following expressions and observe the computed result:

```
4.5 // 2
4 // 2
```

Now consider the following question. What will be the value of k after the following code is executed?

```
x = 3
y = 4

def incremental(x, y=0):
   temp = x**0.5
   temp += y + x**2 + 1.0
   return (temp+5)//2

k = incremental(9,x)
```

- 23.0
- 23
- 46.5
- 92
- 46.0
```

What will be printed on executing the code below?

```
x = "10.0"
w = float(x)
z = w + 10.0
print(type("z"))
```

- 20
- <class 'int'>
- "20.0"
- <class 'float'>
- <class 'str'> ✔
- 20.0

What is the output for the code below?

```
def compare(y1, y2):
   if y1 > y2:
       return y2*y1
   elif y1<=y2:
       return y1+y2
   else:
       return "Not equal"

print(compare(5, 4))
```

- 9
- Not equal
- 20 ✔
- Code throws an error

What will be printed when we execute the code below?

```
i = 1
while i < 5:
    print(i)
    if i % 3 == 0:
        break
i += 1
```

- ```
  1
  2
  3
  ```
- ```
  1
  2
  3
  4
  ```
- ```
  1, 2, 3
  ```
- ```
  1, 2, 3, 4, 5
  ```
- This is an infinite loop printing 1 in every new line ✔
- This is an infinite loop printing natural numbers one in each line

What is printed by the execution of the following code snippet?

```
def f(x=2, y=1):
   print(y, x+1)

def g(x, y):
   h(y**2)

def h(y, x):
   g(x, y=x)

x=10
y=20
f(y, x)
```

- ```
  11 20
  ```
- ```
  21 10
  ```
- ```
  21 11
  ```
- ```
  10 21
  ``` ✔
- ```
  22 11
  ```
  ```

What will be printed by the following sequence of shell commands?

```
echo two > one.txt
echo one > three.txt
echo one.txt > two.txt
cp two.txt three.txt
cp three.txt one.txt
cat three.txt
```

- one
- one.txt ✓
- two.txt
- three.txt
- two

What is printed by the following code?

```
i = 2
s = "2020"

while i < 10:
   i += 1
   if i % 3 == 0:
      s += str(i)
   else:
      break
      s += ","

print(s)
```

- `20203` ✓
- `2020369`
- `20206`
- `"20203"`
- `"20203,"`

You are going to implement the mathematical expression **max(min(num1, num2), num3)** as a python function **max_min**.

Please **select the correct operators from the drop down** menus such that **max_min** implements **max(min(num1, num2), num3)**.

```
def max_min(num1, num2, num3):
    if num1 [1] num2:
        if num2 [2] num3:
            return num2
        else:
            return num3
    elif num1 [3] num3:
        return num1
    else:
        return num3
```

Correct answers:

1 > 2 > 3 >

Consider these functions for the following five questions (assume any variables passed as arguments to these functions contain booleans):

```
def g(x, y):
    if x:
        if y:
            return True
        else:
            return False
    else:
        return False

def h(x, y):
    if x:
        return False
    else:
        if y:
            return False
        else:
            return True
```

What does g(False, False) evaluate to ?

- True
- False ✓

What does h(False, False) evaluate to?

- True ✓
- False

A call to g(b1, b2) will return a result that is equal to which of the following expressions?

Hint: Assume b1 and b2 are of type bool

- b1 != b2
- b1 and b2 ✔
- b1 == b2
- b1 or b2

A call to h(b1, b2) will return a result that is equal to which expression?

Hint: Assume b1 and b2 are of type bool

- b1 and b2
- not (b1 or b2) ✔
- b1 or b2
- not (b1 and b2)

What does the following function call evaluate to?

```
h(g(1>2, 2>1), g(4**0.5==2, False))
```

- True ✔
- False

Pókemon Battle

For the following questions, **total_stat(pkmn)** function will return the total statistics for a given Pókemon. See below definition for **total_stat(pkmn)** function:

```
def total_stat(pkmn):
    return get_hp(pkmn) + 3*get_attack(pkmn) + 3*get_defense(pkmn) + 3*get_sp_atk(pkmn) + 2*get_sp_def(pkmn) + get_speed(pkmn)
```

In the following questions, the APIs get_hp(pkmn), get_attack(pkmn), get_defense(pkmn), get_sp_atk(pkmn), get_sp_def(pkmn), get_speed(pkmn), get_type1(pkmn), get_type2(pkmn) and get_region(pkmn), will repectively return the hp, attack, defense, special attack, special defense, speed, type1, type2 and region of a Pókemon.

There are four Pókemon in our Pókemon dataset. Compute the result of total_stat for each of the Pókemon in the data set. You will need these numbers for the following questions.

Name	HP	Attack	Defense	Special Attack	Special Defense	Speed	Type1	Type1	Region
A	20	25	30	15	10	20	Bug	None	Kalos
B	20	10	15	10	15	30	Fighting	Flying	Alola
C	40	15	15	10	15	50	Fire	Dark	Kanto
D	40	15	20	20	30	30	Flying	Dragon	Kanto

What is the function return of **total_stat('A')**?

- 120
- 250
- 265
- 270 ✔
- 280

The following is the our version of a function for Pókemon battles. Please answer the following questions using the definition given below.

```
def battle_v1(pkmn1, pkmn2) :
    stat1 = total_stat(pkmn1)
    stat2 = total_stat(pkmn2)
    if stat1 > stat2 :
        return pkmn1
    elif stat1 < stat2 :
        return pkmn2
    else:
        if get_speed(pkmn1) > get_speed(pkmn2) :
            return pkmn1
        elif get_speed(pkmn1) < get_speed(pkmn2) :
            return pkmn2
        else:
            return "Draw”
```

What is the function return of **battle_v1(battle_v1("A","B"),battle_v1("C","D"))**? (There is not a Pókemon named "Draw" in our Pókemon database.)

- A
- B
- C
- D ✔
- Draw
- Keyerror : 'Draw'

In this question, we want to rewrite our **battle_v1(pkm1,pkmn2)** to make it shorter. **Drag and drop the right conditional statements** to put them in the correct blocks so that the following code will do exactly what **battle_v1(pkm1,pkmn2)** will do**.**

```
def battle_v1_modified(pkmn1, pkmn2) :
    stat1 = total_stat(pkmn1)
    stat2 = total_stat(pkmn2)

    if [1]                          :
        return pkmn1

    elif [2]                        :
        return pkmn2
    return "Draw"
```

- (get_speed(pkmn1) > get_speed(pkmn2) and (not(stat1 > stat2) and not(stat1 < stat2)) or stat1 > stat2
- (get_speed(pkmn1) < get_speed(pkmn2) and stat1 == stat2) and stat1 < stat2
- (get_speed(pkmn1) > get_speed(pkmn2) and (not(stat1 > stat2) or not(stat1 < stat2)) or stat1 > stat2
- (get_speed(pkmn1) < get_speed(pkmn2) or stat1 == stat2) or stat1 < stat2
- (get_speed(pkmn1) < get_speed(pkmn2) and stat1 == stat2) or stat1 < stat2
- (get_speed(pkmn1) > get_speed(pkmn2) and (not(stat1 > stat2) and not(stat1 < stat2)) and stat1 > stat2

Correct answers:

1. (get_speed(pkmn1) > get_speed(pkmn2) and (not(stat1 > stat2) and not(stat1 < stat2)) or stat1 > stat2
2. (get_speed(pkmn1) < get_speed(pkmn2) and stat1 == stat2) or stat1 < stat2

Battleship

Read the code. This code is suppose to draw a Battleship board as developed during lecture with an 'X' at the coordinates entered by the user. **However, this code contains a semantic error and does not print the 'X'.**

```
#1  def draw (m=4, n=2, M=6, N=8):
#2    i = 0
#3     while i < M:
#4        j = 0
#5        while j < N:
#6           if m == i and n == j:
#7              print("X", end="")
#8           else:
#9              print(".", end="")
#10           j += 1
#11       print() # just a newline because end="\n" by default
#12       i += 1

#13  x = input("Enter x: ")
#14  y = input("Enter y: ")
#15  draw(x,y)
```

We intended for this code to display X at the input coordinates. However, if you run the code with input x as 1 and y as 2, no X will be printed on the board. Select the best option to explain why?

- the print() function is used incorrectly
- the input coordinates are out of the range of the map
- parameters are passed incorrectly to draw()
- the type of x and y are not of type int ✔

For the following questions assume the bug in the battleship function has been fixed and the code correctly prints the X.

What input would print the map with an X in the top-right corner?

Enter x: 1

Enter y: 2

Correct answers:

1 0 2 7

Which parameter to draw() represents the width of the map?

- m
- n
- M
- N ✓

Which of the following lines will print the same thing as draw(-1, -1, M=4, N=8) and input x as -1, y as -1 prints the equivalent of what?

- print(("."*4 + "\n") * 8)
- print("."*4*8)
- print(("."*8 + "\n") * 4 + "X")
- print(("."*8 + "\n") * 4) ✓
- print("X" + ("."*8 + "\n") * 4)

What is the output of draw(2), assuming the code can print X correctly. ?

-
```
......
.X....
......
......
......
......
```

-
```
.X....
......
......
......
......
......
```

-
```
......
...X..
......
......
......
......
```

- ✓
```
......
......
..X...
......
......
......
```

Choose the correct option from the drop down to compute the average police spending over the period from 2015 to 2020. **There are two answers that produce the correct value; you should choose the better one that avoids hardcoding.**

```
total = 0
start = 2015
end = 2020
year = start
# assume get_budget returns the spending by an agency in a given year
while year <= end:
    total += get_budget("police", year)
    year += 1
avg = total / [1]
```

Correct answers:

1 (end - start + 1)

What is the type of the variable "**year**" in the code above?

- bool
- int ✓
- float
- str

What is the output after executing the following block of code?

```
def red() :
    blue(1)
    black(2)
    purple(3)

def blue(x):
    print(x, end=" ")

def black(y):
    blue(y + 2)
    print(y, end=" ")

def purple(z) :
    black(z)

red()
```

- 1 2 4 5 3
- 1 4 2 7 5
- 2 4 7 5 1
- 1 4 2 5 3 ✓

```
n = 6
isOver = False
while ((n > 0) or (isOver == False)) :
    if (n > 0) :
        print(n)
        n = n - 1
    else:
        isOver = True
```

Which of the following statements about the above code are **TRUE,** select all that apply?

- [x] There are 3 code blocks
- [ ] The statement while ((n > 0) or (isOver == False)) will be executed 6 times.
- [x] The loop will terminate if we set n to 0 instead of 6 in the first line.
- [ ] The output after executing will be as follows:

```
6
5
4
3
2
1
0
```

What is the output after executing the following block of code?

```
bookSalesToday = 200
bookSalesYesterday = 230

def salesDiff(day1, day2 = 0):
    difference = day2 - day1
    return difference

salesDiff(bookSalesToday)
```

- (x) -200
- ( ) -30
- ( ) 0
- ( ) 30
- ( ) 200
- ( ) 230

How many times will the condition i < 7 be evaluated?

```
i = 2
while i < 7:
    if i == 3:
        i = i + 2
    else:
        i = i + 1
```

- 1
- 2
- 3
- 4
- 5 ✔
- 6
- 7

What is the output after executing the following block of code?

```
def z(x, y):
    a = 3
    b = 8
    return x + y

a = 7
b = 2
print(z(a, b)+b)
```

- 11 ✔
- 13
- 17
- 19

If we modify the above code as follows, what is the output after executing this modified block of code?

```
def z(x, y) :
    global a
    global b
    a = 3
    b = 8
    return x + y

a = 7
b = 2
print(z(a, b) + b)
```

- 11
- 13
- 17 ✔
- 19