Oblig 5 – Programmering 1.

Oppgave 1.1

```
a = 4
b = 2
c = (a + b) / b

print(c ** 2)
```

This prints: 9

Oppgave 1.2

```
Numbers = []

for x in range(5):
        numbers.append(x)

print(numbers)
```

This prints: [0, 1, 2, 3, 4]

For loop and for range in 5, in this instance, starts at 0 and prints from 0 to 5, but not including 5. Important to remember.

```
Numbers = []

for x in range(5):
        numbers.insert(x, 0)

print(numbers)
```

This prints: [0, 0, 0, 0, 0]

Insert method is used to insert an item at a specified index in a list. The method takes 2 arguments: the first is the index position in the list where you want to place your new item.
Second argument is the actual item you want to insert. So in this instance when we have a for loop (5), loop starts and 0, meaning it will insert to indexpos. 0 in

the list, and then index 1, and so on, until it runs 5 times and last insert is to indexpos. 4 in the list. This creates a list with five zeros.

Oppgave 1.3

```
class Course:
    new *
    def __init__(self, name, number_of_students, study_points=10):
        self.name = name
        self.number_of_students = number_of_students
        self.study_points = study_points

    1 usage  new *
    def get_description(self):
        return f"The course {self.name} has {self.number_of_students} students" \
               f" and {self.study_points} study points."


programmering_1 = Course( name: "Programmering 1",  number_of_students: 215)

print(programmering_1.get_description())
```

This prints: The course Programmering 1 has 215 students and 10 study points.

```
#  in OOP (object-oriented programming) objects are instances of classes. A
class is like a blueprint,
#  and an object is an actual entity created based on that blueprint.

# A method is a function that is defined inside a class and is meant to be
called on
# instances of that class, that is, Objects.
# The syntax for calling a method on an object can be described with the
example:
# print(programmering_1.get_description())
```

Oppgave 1.4

```
animals_in_zoo = {"honey badger": 2, "ape": 15, "zebra": 6, "giraffe": 4}  # creating a dictionary with zoo animals

for animal in animals_in_zoo:
    if animals_in_zoo[animal] < 5:
        print(animal.title())
```

This prints: Honey Badger and Giraffe

Title method in python converts a string to what is called title case, where the first letter of each word in the string is capitalized, and the rest are lowercased.

Oppgave 1.5

```python
numbers = [5, 2, 3, 2, 4, 1]

sum_of_numbers = 0
for number in numbers:
    if number <= 2:
        sum_of_numbers += number
    else:
        sum_of_numbers += 1

print(sum_of_numbers)  # prints 8
```

This prints: 8

Oppgave 1.6

```python
animals = ["honey badger", "giraffe", "ape", "zebra"]

animals[1] = "elephant"

animals.sort()

animals = animals[:2]

for animal in animals:
    print(animal)  # print will be ape and elephant
```

This prints: ape and elephant.

[:2] slices the list and keeps only the first two elements (at index 0 and 1).
So basically this syntax means “start from the beginning of the list and go up to (but not including) the element at index 2.

Oppgave 1.7

```python
animals = ["Elephant", "Dog", "Cat", "Gorilla", "Dodo"]

animals = animals[1:3]  # slices list and only takes with it items from index 1 to index 2 (not including 3)
#  List now consists of only dog and cat, meaning the new list has 2 index positions, index 0 and 1.

animals[0] = "Alligator"
# Dog becomes Alligator

animals.sort(reverse=True)
# Alligator and cat becomes Cat and alligator

print(animals)  # prints cat and alligator
```

This prints: cat and alligator

Oppgave 1.8

```python
shopping_list = {}


5 usages  new *
def add_item(item_name, quantity=1):
    if item_name in shopping_list.keys():
        shopping_list[item_name] += quantity

    else:
        shopping_list[item_name] = quantity  # The syntax dictionary[key] = value is how you assign a value to
        # a key in a dictionary.
```

This prints: Bread: 4, Milk: 3, Eggs: 1

Oppgave 1.9

```python
x = 0

for i in range(0, 5, 2):
    x += i

print(x)

```

This prints in first instance: 6

```python
x = 0

for i in range(0, 5):
    x += i

print(x)
```

This prints in second instance: 10

Oppgave 1.10

```python
a = 5
b = 2
c = 0

c += a ** b
print(c)  # prints 25

c += a % b
print(c)  # prints 26

c += a - b * 2
print(c)  # prints 27

c //= b  # // floor division, rounds down to nearest whole number
print(c)  # prints 13
```

Prints in order: 25, 26, 27 and lastly 13

Oppgave 1.11

```
2 usages  new *
class Game:
    new *
    def __init__(self, name, genre, age_rating=18):
        self.name = name
        self.genre = genre
        self.age_rating = age_rating

    1 usage  new *
    def description(self):
        return f"The game {self.name} is of the genre {self.genre} and has an age rating of {self.age_rating}"


game1 = Game( name: "Hades",  genre: "Rogue-lite",  age_rating: 12)
game2 = Game( name: "God of War",  genre: "Action")
print(game1.age_rating)
print(game2.description())
```

First print will be: 12
Second print will be: The game God of War is of the genre Action and has an age rating of 18

Oppgave 1.12

```
randomList = [1, "a", 2, "b", 3]
result = 0

for entry in randomList:
    try:
        result += int(entry)
    except ValueError:
        print("A ValueError occurred.")

print(f"The sum is {result}")  # Result will be 6, "a" and "b" will print the ValueError

```

This will print: ValueError, ValueError, 6