Facit och kommentarer till prov 2019-04-17 i DA2004

Del A: flervalsfrågor

1. *D*
2. *A*
3. *A*,*E*
4. *D*
5. *C*
6. *C*
7. *B*
8. *A*

Del B: kodfrågor

9.
```
def f(s):
    s_list = []
    for item in s
        s_list.append(item)
    return s_list
```

10. Möjlig lösning:

```
def my_div_calc(l, x):
    counter = 0
    for item in l:
        try:
            n = float(item)
        except:
            continue

        if n % x == 0:
            counter += 1
    return counter
```

11.
```
def add_up(lis):
    if len(lis)==0:
        return 0
    else:
        elem = lis.pop()
        return elem + add_up(lis)
```

    eller

```
def add_up(lis):
    if len(lis)==1:
        return lis.pop()
    else:
        elem = lis.pop()
        return elem + add_up(lis)
```

12. Funktionen tar summan av tal i z2 och kvadrerade tal i z1 och summerar upp.

```
def f2(z1,z2):
    s = 0
    n = min(len(z1),len(z2))
    for i in range(n):
        s += z1[i]**2 + z2[i]
    return s
```

Alternativ med zip:

```
def f3(z1, z2):
    s = 0
    for a, b in zip(z1, z2):
        s += a**2 + b
    return s
```

13.
```
c1 = Circle(2,3,3)
c2 = Circle(0,1,4)
c1 < c2 # eller c1.__lt__(c2)
```

14.
```
    def __eq__(self, other): # compare area
        return self.r == other.r and  self.x == other.x and  self.
            y == other.y
```

15. Exempel

```
class Circle:
    def __init__(self, x, y, r):
        self.x = x
        self.y = y
        self.r = r

    def __lt__(self, other): # compare area
        return self.r < other.r

    def __eq__(self, other):
        return self.r == other.r and  self.x == other.x and  self.
            y == other.y

    def is_inscribed(self, other):
        if self.r > other.r: # definitely not inscribed
            return False
        elif self.r == other.r: # Only inscribed if identical
            return  self.x == other.x and self.y == other.y

        else: # we know radius of self is smaller
            # get distance between centers using pythagoras
                theorem
            center_diff = ((self.x - other.x)**2 + (self.y - other
                .y)**2)**0.5
            return center_diff + self.r <= other.r

c1 = Circle(0,0,1)
c2 = Circle(1,2,3)
print(c1.is_inscribed(c2))
```

16. Möjlig lösning:

```
def check_presence(w, T):
    curr_pos = 0
    for l in w:
        l_found = False
        for i in range(curr_pos, len(T)):
            if l == T[i]:
                l_found = True
                curr_pos = i
                break
        if not l_found:
            return False
    return True
```