Lösung zur Klausur #1982

Es wird keine Garantie für die Richtigkeit gegeben. Diese Lösung ist von Studenten angefertigt.

Aufgabe 1

a)

```
>>> 'I %s %sam' % ('spam'[2:], 'spam'[:2])
'I am spam'
```

b)

```
>>> a=((i, j) for i in range(1, 4) for j in range(1, 7, 2)
...            if i**2 < j*i)
>>> tuple(a)
((1, 3), (1, 5), (2, 3), (2, 5), (3, 5))
```

c)

```
>>> import re
>>> re.findall(r'([a-z]+?)\w*', "Ham, spam, and, eggs")
['a', 's', 'a', 'e']
```

d)

```
>>> from operator import itemgetter
>>> a = [3, 2, 1]
>>> b = ("a", "C", "b")
>>> sorted(zip(a, b), key=itemgetter(1), reverse=True)
[(1, 'b'), (3, 'a'), (2, 'C')]
```

e)

```
>>> a, b = (3, 2, 1, 0), (2, 1 , 3, 4, 5)
>>> list(map(max, a, b))
[3, 2, 3, 4]
```

f)

```
>>> f = lambda y: lambda x: int(x - y/2)
>>> f(-56)(14)
42
```

Aufgabe 2

a)

- statische Finitheit
- dynamische Finitheit
- Effektivität
- Präzise
- Terminiert

b)

imperative Programmierung: Man beschreibt, wie etwas erreicht werden soll
deklarative Programmierung: Man beschreibt, was erreicht werden soll.

c)

Imperativ	Deklarativ
prozedurale Programierung OOP	funktionale Programmierung

d)

- 1936 Turing-Maschine
- 1941 Erfindung des Z3
- 1948 Erster Speicherprogrammierbarer Computer
- 1971 Erster Mikroprozessor
- 1989 Beginn der Entwicklung des WWW

Aufgabe 3

a)

```
def sieve(n):
    siv=set(range(2,n+1))
    for i in range(2, n+1):
        if i in siv:
            for x in range(i**2, n+1):
                if x % i == 0:
                    siv.discard(x)
    return siv

sieve(20)
```

b)

```
def isprime(n):
    if n == 0 or n==1:
        return False
    if n in sieve(n):
        return True
    else:
        return False

isprime(20)
```

Aufgabe 4

a)

```
def reachable(sender, receiver, confidants):
    marked = dict((x, False) for x in confidants)
    marked[sender] = True
    L = [sender]
    while L:
        print(marked, "\n" , L, "\n")
        cur = L.pop()
        if cur == receiver:
            return True
        for f in confidants[cur]:
            if not marked[f]:
                marked[f] = True
                L.append(f)
    return False
```

```
cnfd = {1: [2, 3], 2: [5, 4], 3: [3], 4: [6, 3], 5: [6], 6:[5]}
reachable(1, 6, cnfd)
```

Ausgabe:

```
{1: True, 2: False, 3: False, 4: False, 5: False, 6: False}
 [1]

{1: True, 2: True, 3: True, 4: False, 5: False, 6: False}
 [2, 3]

{1: True, 2: True, 3: True, 4: False, 5: False, 6: False}
 [2]

{1: True, 2: True, 3: True, 4: True, 5: True, 6: False}
 [5, 4]

{1: True, 2: True, 3: True, 4: True, 5: True, 6: True}
 [5, 6]
```

`True`

b)

$O(n^2)$

Aufgabe 5

a)

```
def fractions():
    zaehler, nenner, direction = 1, 1, "up"
    while True:
      yield zaehler, nenner
        if zaehler == 1:
            if direction == "up":
                nenner += 1
                direction = "down"
            else:
                zaehler += 1
                nenner -= 1
        elif nenner == 1:
            if direction == "down":
                zaehler += 1
                direction = "up"
            else:
                zaehler -= 1
                nenner += 1
        elif direction == "up":
```

```
            zaehler -= 1
            nenner += 1
        else:
            zaehler += 1
            nenner -= 1
```

b)

```
def print_fractions(n):
    f = fractions()
    for i in range(n):
        print(next(f))
```

Aufgabe 6

```
user = 'Alice'
def decorator(f):
    user = 'Bob'
    def wrapper(*args, **kwargs):
        global user
        res = '%s %s' % (user, f(*args, **kwargs))
        return res
    return wrapper

@decorator
def func1(str):
    return '%s im Hoersaal.' % str

def func2(str):
    return '%s %s ein Buch.' % (user, str)

print(func1('schreibt'))
print(func2('liest'))
```

a)

```
Bob schreibt im Hoersaal.
Alice liest ein Buch.
```

b)

```
Alice schreibt im Hoersaal.
Alice liest ein Buch.
```

Aufgabe 7

a)

```
class Marmelade:
    step = 5
    def __init__(self, start, limit):
        self.run = start
        self.limit = limit

    def __iter__(self):
        return self

    def __next__(self):
        self.run += self.step
        if self.run > self.limit:
            raise StopIteration
        return self.run

jam = Marmelade(32, 45)
for i in jam:
    print(i, end=', ')

for i in jam:
    print(2*i, end=', ')

print("Empty!")
```

b)

`37, 42, Empty`

c)

Iteratorprotokoll

Aufgabe 8

a)

`afunc`

$$\prod_{i=k}^{n} f(i)$$

`bfunc`

$$\prod_{i=1}^{n} 2 \cdot n$$

b)

```
def cfunc(f, k, n):
    if k=n+1:
        return 1
    else:
        return f(n)*cfunc f,k,n-1
```