Организованное
место Псевдофиз Наук

Библиотека незгораемая

Super Hot

Прикладное Ахтоведение

справочник
персонажа-лучника

Издательство «светлое кольцо»

Нуарсити

2007

Содержание

```
    7. Сохраняем
    """
    args = init_args(sys.argv)
    pdf_in = os.path.abspath(args.pdf_in[0])
    pdf_out = os.path.abspath(args.pdf_out[0])
    rotate180_even = args.rotate180_even
    out_one_by_one = args.out_one_by_one
    save_odd_even = args.save_odd_even
    add_blank_before, add_blank_after = args.add_blank_before, args.add_blank_after
    dpi = args.dpi

    pout(f"Getting from pdf \"{pdf_in}\" pages with dpi={dpi}... " , endl=False)
    imgs = pdf_convert_from_path(pdf_in, dpi)
    pout("OK")
    if(len(imgs) <= 0):
        print(f"\"pdf_in\" has {len(imgs)} pages. Exitting...")
        exit()

    imgs = solve_transform(imgs, args)

    imgs = adding_blank(imgs, add_blank_before, add_blank_after)

    imgs = fill_4(imgs)

    imgs_odd, imgs_even = split(imgs)

    imgs_odd = A5_A4(imgs_odd, False)
    imgs_even = A5_A4(imgs_even, True)

    if(rotate180_even == True):
        imgs_even = rotate_even_180(imgs_even)

    if(out_one_by_one == False):
        pout("Connecting the pages together: first the numbers with odd, then with even... ", endl=False)
        imgs = imgs_odd + imgs_even # insert
        pout("OK")
    else:
        pout(f"Connecting pages together: odd ({len(imgs_odd)}) and even ({len(imgs_even)}) pages alternate
one by one... ", endl=False)
        imgs = [imgs_odd[i//2] if i%2==0 else imgs_even[i//2] for i in range(len(imgs_odd)+len(imgs_even))]
        pout("OK")

    pout(f"Saving output pdf \"{pdf_out}\"... ", endl=False)
    imgs[0].save(pdf_out, save_all=True, append_images=imgs[1:])
    pout("OK")

    if(save_odd_even == True):
        fn, fe = os.path.splitext(pdf_out)
        odds_path, evens_path = fn + "_odds" + fe, fn + "_evens" + fe
        pout(f"Saving odds \"{odds_path}\"... ", endl=False)
        imgs_odd[0].save(odds_path, save_all=True, append_images=imgs_odd[1:])
        pout("OK")
        pout(f"Saving odds \"{evens_path}\"... ", endl=False)
        imgs_even[0].save(evens_path, save_all=True, append_images=imgs_even[1:])
        pout("OK")


    pout(f"{'='*20}DONE!{'='*20}")
```

Bee Movie Script

According to all known laws

of aviation,

there is no way a bee

should be able to fly.

Its wings are too small to get

its fat little body off the ground.

The bee, of course, flies anyway

because bees don't care

what humans think is impossible.

Yellow, black. Yellow, black.

Yellow, black. Yellow, black.

Ooh, black and yellow!

Let's shake it up a little.

В необъятных просторах лунной пустоши, где магический свет покрывает землю, сто
существо, которое отличалось своей невероятной эксцентричностью. Это было нечт
среднего между космической курицей и пурпурным единорогом. У него были перья
разных цветов, меняющихся в зависимости от его настроения, а его рог пронизывал

```python
# -*- coding: utf-8 -*-

import argparse
import sys
import os

from pdf2image import convert_from_path as pdf_convert_from_path

from img_hundler import ImageHundler

def init_args(args: list) -> "ArgumentParser":
    parser = argparse.ArgumentParser(prog = "mkbook",
        description="Make book from pdf to print.")
    parser.add_argument("pdf_in", type=str, nargs=1,
                        help="Path to source pdf.")
    parser.add_argument("pdf_out", type=str, nargs=1,
                        help="Path to output pdf")
    parser.add_argument("--dpi", type=int, default=200, required=False,
                        help="More value means better quality, but larger file size.")
    parser.add_argument("--rotate_left", default=True, action='store_false',
                        help="When rotating landscape (horizontally) sheets, rotate them to the left
(counterclockwise).")
    parser.add_argument("--rotate180_even", default=False, action='store_true',
                        help="Rotates output pages with even numbers by 180 degrees.")
    parser.add_argument("--out_one_by_one", default=False, action='store_true',
                        help="The output pages go one after the other, not the even ones first, and then the
odd ones.")
    parser.add_argument("--save_odd_even", default=False, action='store_true',
                        help="Additionally save pages with odd and even numbers separately.")
    parser.add_argument("--add_blank_before", type=int, default=0,
                        help="Add blank sheets to the beginning.")
    parser.add_argument("--add_blank_after", type=int, default=0,
                        help="Add blank sheets to the end.")
    args = parser.parse_args(args[1:])

    return args
```

```python
def solve_transform(imgs: list, args: list) -> list:
    left_right = not args.rotate_left
    pout("Resizing for A4 and landscape to portrait... ", endl=False)
    for i, img_i in enumerate(imgs):
        new_img = ImageHundler(img_i)
        new_img.resize_4A4_fill()
        img_size = new_img.get().size
        if(img_size[0] > img_size[1]): # Landscape
            new_img.rotate90(left_right)
        imgs[i] = new_img.get()
    pout("OK")

    pout("Bringing all pages to the same size... ", endl=False)
    max_w = max(imgs, key=lambda x:x.size[0]).size[0]
    max_h = int(max_w* (2**(0.5)) + 0.5)
    res = []
    for img_i in imgs:
        buff = ImageHundler(img_i)
        buff.resize((max_w, max_h))
        res.append(buff.get())
    pout("OK")

    return res

def adding_blank(imgs: list, before: int, after: int) -> list:
    blank = ImageHundler(imgs[0])
    blank = blank.get_as_empty()

    res_before = []
    for i in range(before):
        res_before.append(blank.copy())

    res_after = []
    for i in range(after):
        res_after.append(blank.copy())
```

Barry! Breakfast is ready!

Ooming!

Hang on a second.

Hello?

- Barry?

- Adam?

- Oan you believe this is happening?

- I can't. I'll pick you up.

Looking sharp.

Use the stairs. Your father

paid good money for those.

Sorry. I'm excited.

Here's the graduate.

We're very proud of you, son.

A perfect report card, all B's.

Very proud.

Ma! I got a thing going here.

- You got lint on your fuzz.

- Ow! That's me!

- Wave to us! We'll be in row 118,000.

- Bye!

Barry, I told you,

stop flying in the house!

- Hey, Adam.

- Hey, Barry.

- Is that fuzz gel?

- A little. Special day, graduation.

Never thought I'd make it.

Three days grade school,

three days high school.

Those were awkward.

Three days college. I'm glad I took

a day and hitchhiked around the hive.

You did come back different.

- Hi, Barry.

- Artie, growing a mustache? Looks good.

- Hear about Frankie?

- Yeah.

- You going to the funeral?

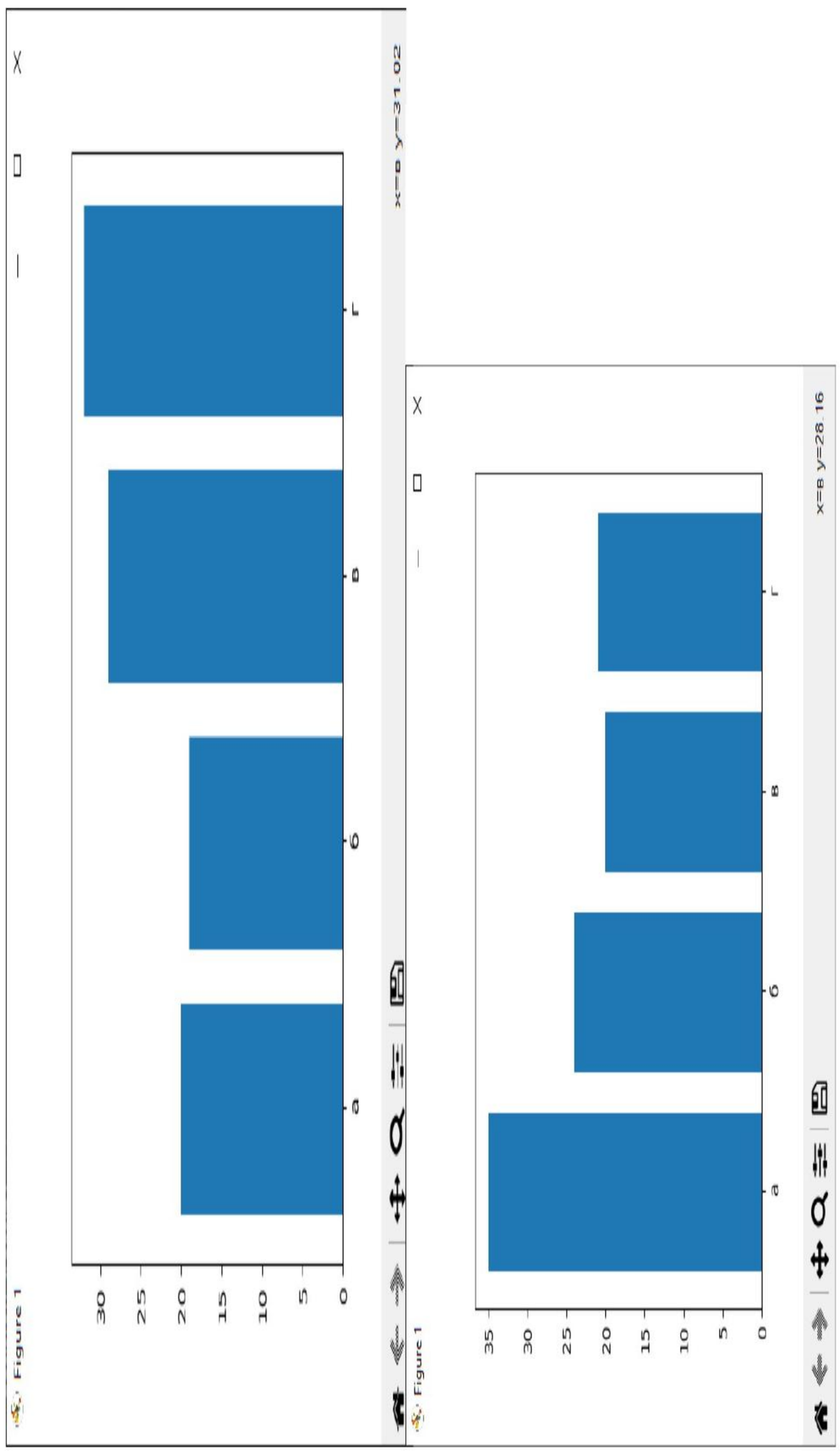
- No, I'm not going.

Everybody knows,

sting someone, you die.

Don't waste it on a squirrel.

```
    return res_before + imgs + res_after

def fill_4(imgs: list) -> list:
    c = len(imgs)
    if(c % 4 == 0):
        return imgs
    else:
        pout("Adding blank pages so that their number is a multiple of 4... ", endl=False)
        buff = ImageHundler(imgs[0])
        diff = 4 - (c % 4)
        res = imgs + [buff.get_as_empty() for _ in range(diff)]
        pout("OK")
        return res

def split(imgs: list) -> "([odd], [even])":
    pout("Dividing pages into 2 sets: odd and even... ", endl=False)
    odd, even = [], []
    for i, img_i in enumerate(imgs):
        if((i+1)%2 == 1): # odd
            odd.append(img_i)
        else: # even
            even.append(img_i)
    pout("OK")
    return (odd, even)

def rotate_even_180(imgs: list) -> list:
    pout(f"Rotating pages with even numbers 180 degrees... " , endl=False)
    for i, img_i in enumerate(imgs):
        imgs[i] = imgs[i].rotate(180, expand=True)
    pout("OK")
    return imgs

def A5_A4(imgs: list, odd_even: bool = False) -> list:
    """
    odd_even == False -> odd
    odd_even == True -> even
    """
    odd_even_str = "odd" if odd_even == False else "even"
    pout(f"Fastening the pages in a set of {odd_even_str} pages... ", endl=False)
    c = len(imgs)
    if(c % 2 != 0):
        raise ValueError(f"Len ({c}) of imgs must be even! ")

    res = []
    for i in range(c//2):
        img_l, img_r = imgs[i], imgs[c-1-i]
        if(odd_even == False): # !!!
            img_l, img_r = img_r, img_l
        img_l = ImageHundler(img_l)
        img_l.append_2right(img_r)
        res.append(img_l.get())

    pout("OK")
    return res

def pout(s: str, endl: bool = True):
    if(endl==False):
        print(s, end="")
    else:
        print(s)
    sys.stdout.flush()

if __name__ == '__main__':
    """
    1. Получить все листы из pdf, как png.
    2. Сделать их все формата АХ (1:1.41).
    3. Сделать все листы портретными.
    4. Добавить пустых листов, чтобы их кол-во было кратно 4.
    5. Делаем 2 множества: чётных и нечётных.
    6. Соединяем (append_2right) элементы этих множеств.
```



Such a hothead.

I guess he could have

just gotten out of the way.

I love this incorporating

an amusement park into our day.

That's why we don't need vacations.

ПЕРИОДИЧЕСКАЯ СИСТЕМА ХИМИЧЕСКИХ ЭЛЕМЕНТОВ Д. И. МЕНДЕЛЕЕВА

ПЕРИОДЫ	ГРУППЫ ЭЛЕМЕНТОВ							
	A I B	A II B	A III B	A IV B	A V B	A VI B	A VII B	A VIII B
1	(H)						1 **H** 1.00794 Hydrogenium **Водород**	2 **He** 4.002602 Helium **Гелий**
2	3 **Li** 6.941 Lithium **Литий**	4 **Be** 9.0122 Beryllium **Бериллий**	5 **B** 10.811 Borum **Бор**	6 **C** 12.011 Carboneum **Углерод**	7 **N** 14.007 Nitrogenium **Азот**	8 **O** 15.999 Oxygenium **Кислород**	9 **F** 18.998 Fluorum **Фтор**	10 **Ne** 20.179 Neon **Неон**
3	11 **Na** 22.99 Natrium **Натрий**	12 **Mg** 24.305 Magnesium **Магний**	13 **Al** 26.9815 Aluminium **Алюминий**	14 **Si** 28.086 Silicium **Кремний**	15 **P** 30.974 Phosphorus **Фосфор**	16 **S** 32.066 Sulfur **Сера**	17 **Cl** 35.453 Chlorium **Хлор**	18 **Ar** 39.948 Argon **Аргон**
4	19 **K** 39.098 Kalium **Калий**	20 **Ca** 40.08 Calcium **Кальций**	21 **Sc** 44.956 Scandium **Скандий**	22 **Ti** 47.90 Titanium **Титан**	23 **V** 50.941 Vanadium **Ванадий**	24 **Cr** 51.996 Chromium **Хром**	25 **Mn** 54.938 Manganum **Марганец**	26 **Fe** 55.847 Ferrum **Железо**	27 **Co** 58.933 Cobaltum **Кобальт**	28 **Ni** 58.70 Niccolum **Никель**
	29 **Cu** 63.546 Cuprum **Медь**	30 **Zn** 65.39 Zincum **Цинк**	31 **Ga** 69.72 Gallium **Галий**	32 **Ge** 72.59 Germanium **Германий**	33 **As** 74.992 Arsenicum **Мышьяк**	34 **Se** 78.96 Selenium **Селен**	35 **Br** 79.904 Bromum **Бром**	36 **Kr** 83.80 Krypton **Криптон**		
5	37 **Rb** 85.468 Rubidium **Рубидий**	38 **Sr** 87.62 Strontium **Стронций**	39 **Y** 88.906 Yttrium **Иттрий**	40 **Zr** 91.22 Zirconium **Цирконий**	41 **Nb** 92.906 Niobium **Ниобий**	42 **Mo** 95.94 Molybdaenum **Молибден**	43 **Tc** 97.91 Technetium **Технеций**	44 **Ru** 101.07 Ruthhenium **Рутений**	45 **Rh** 102.906 Rhodium **Родий**	46 **Pd** 106.4 Palladium **Палладий**
	47 **Ag** 107.868 Argentum **Серебро**	48 **Cd** 112.41 Cadmium **Кадмий**	49 **In** 114.82 Indium **Индий**	50 **Sn** 118.71 Stannum **Олово**	51 **Sb** 121.75 Stibium **Сурьма**	52 **Te** 127.60 Tellurium **Теллур**	53 **I** 126.9045 Iodum **Иод**	54 **Xe** 131.29 Xenon **Ксенон**		
6	55 **Cs** 132.905 Cesium **Цезий**	56 **Ba** 137.33 Barium **Барий**	57 **La*** 138.9055 Lanthanum **Лантан**	72 **Hf** 178.49 Hafnium **Гафний**	73 **Ta** 180.9479 Tantalum **Тантал**	74 **W** 183.85 Wolframium **Вольфрам**	75 **Re** 186.207 Rhenium **Рений**	76 **Os** 190.2 Osmium **Осмий**	77 **Ir** 192.22 Iridium **Иридий**	78 **Pt** 195.08 Platinum **Платина**
	79 **Au** 196.967 Aurum **Золото**	80 **Hg** 200.59 Hydrargyrum **Ртуть**	81 **Tl** 204.38 Thallium **Таллий**	82 **Pb** 207.19 Plumbum **Свинец**	83 **Bi** 208.980 Bismuthum **Висмут**	84 **Po** 209.98 Polonium **Полоний**	85 **At** 209.99 Astatium **Астат**	86 **Rn** [222] Radon **Радон**		
7	87 **Fr** [223] Francium **Франций**	88 **Ra** [226] Radium **Радий**	89 **Ac**** [227] Actinium **Актиний**	104 **Rf** [261] Rutherfordium **Резерфордий**	105 **Db** [262] Dubnium **Дубний**	106 **Sg** [263] Seaborgium **Сиборгий**	107 **Bh** [262] Bohrium **Борий**	108 **Hs** [265] Hassium **Хассий**	109 **Mt** [266] Meitnerium **Мейтнерий**	110 [269]
ФОРМУЛЫ ВЫСШИХ ОКСИДОВ	R_2O	RO	R_2O_3	RO_2	R_2O_5	RO_3	R_2O_7	RO_4		
ФОРМУЛЫ ЛЕТУЧИХ ОДНОРОДНЫХ СОЕДИНЕНИЙ				RH_4	RH_3	RH_2	RH			

ЛАНТАНОИДЫ*	58 **Ce** 140.12 Cerium **Церий**	59 **Pr** 140.908 Praseodymium **Празеодим**	60 **Nd** 144.24 Neodymium **Неодим**	61 **Pm** 144.91 Promethium **Прометий**	62 **Sm** 150.36 Samarium **Самарий**	63 **Eu** 151.96 Europium **Европий**	64 **Gd** 157.25 Gadolinium **Гадолиний**	65 **Tb** 158.926 Terbium **Тербий**	66 **Dy** 162.50 Dysprosium **Диспрозий**	67 **Ho** 164.930 Holmium **Гольмий**	68 **Er** 167.26 Erbium **Эрбий**	69 **Tm** 168.934 Thulium **Тулий**	70 **Yb** 173.04 Ytterbium **Иттербий**	71 **Lu** 174.967 Lutetium **Лютеций**
АКТИНОИДЫ**	90 **Th** Thorium **Торий**	91 **Pa** Protactinium **Протактиний**	92 **U** Uranium **Уран**	93 **Np** Neptunium **Нептуний**	94 **Pu** Plutonium **Плутоний**	95 **Am** Americium **Америций**	96 **Cm** Curium **Кюрий**	97 **Bk** Berkelium **Берклий**	98 **Cf** Californium **Калифорний**	99 **Es** Einsteinium **Эйнштейний**	100 **Fm** Fermium **Фермий**	101 **Md** Mendelevium **Менделевий**	102 **No** Nobelium **Нобелий**	103 **Lr** Lawrencium **Лоуренсий**

Символ элемента
Относительная атомная масса
Порядковый номер
Ar 18 39.948 Argon **Аргон**
Название элемента
Распределение электронов на энергетических уровнях