**УСЛОВИЯ ОФОРМЛЕНИЯ**

1. **Поля: слева 3,0, справа 1,5, сверху и снизу 2,0**
2. **Times New Roman**
3. **14 шрифт**
4. **Выравнивание по ширине**
5. **Межстрочный интервал 1,5**
6. **Абзацный отступ 1,25 (не распространяется на рисунки)**
7. **Нумерация, начиная с практической части (5 страница), но учитывая титульные листы. На титульных листах ее быть не должно!**

**Что нужно сделать:**

1. **Отсканировать титульные листы (в следующем порядке: титул, задание, инструктаж, рабочий график) в формате pdf.**
2. **Удалить эту страницу и заполнить отчет далее по образцу согласно требованиям оформления. Убедиться, что нумерация начинается с 5 страницы. Сохранить в pdf.**
3. **Объединить титульные листы и отчет в единый pdf файл** [**https://www.ilovepdf.com/merge\_pdf**](https://www.ilovepdf.com/merge_pdf)
4. **Назвать файл строго следующим образом: ОП\_Фамилия\_Группа.pdf**

**(например, ОП\_Иванов\_ИКБО-30-21.pdf)**

1. **Загрузить отчет в СДО в раздел БАК\_Отчет по практике**

[**https://online-edu.mirea.ru/course/view.php?id=3367**](https://online-edu.mirea.ru/course/view.php?id=3367)

**1) Тема «Основные конструкции языка, коллекции»**

**Задача 1**

Условие задачи: Написать функцию balanced\_num, которая определяет является ли заданное число сбалансированным, т.е. сумма цифр справа и слева от середины равны (abcde ==> a + b == d + e; abcdef ==> a + b == e + f)

Решение задачи:

|  |
| --- |
| def balanced\_num(number):  number = str(number)  n = int((len(number)-1)/2)  a = 0  for i in range(n):  a += int(number[i])  a -= int(number[l en(number)-i-1])  return not a |

Тестирование:

|  |  |  |  |
| --- | --- | --- | --- |
| № | Тест | Ожидаемое значение | Полученное значение |
| 1 | 13 | True | True |
| 2 | 0 | True | True |
| 3 | 295591 | False | False |
| 4 | 56239814 | True | True |
| 5 | 1230987 | False | False |

**Задача 2**

Условие задачи: Задан список положительных чисел arr и положительное число k. Написать функцию int\_diff, которая определяет сколько раз разница между двумя различными числами из списка равна k

Решение задачи:

|  |
| --- |
| def int\_diff(arr, k):  awns = 0  for i in range(len(arr)):  for i2 in range(i+1, len(arr)):  if(abs(arr[i] - arr[i2]) == k): awns+=1  return awns |

Тестирование:

|  |  |  |  |
| --- | --- | --- | --- |
| № | Тест | Ожидаемое значение | Полученное значение |
| 1 | [1, 1, 5, 6, 9, 16, 27], 4 | 3 | 3 |
| 2 | [1, 1, 3, 3], 2 | 4 | 4 |

**Задача 3**

Условие задачи: Некоторые люди достигая определенного возраста, в шутку празднуют только 20 или 21 день рождения, навсегда. С некоторыми математическими навыками это вполне возможно - нужно только выбрать правильную систему счисления. Написать функцию happy\_birthday(age), которая возвращает систему счисления, в которой данный возраст выглядит как 20 или 21.

Решение задачи:

|  |
| --- |
| def fastCheck(num, ss):  return (num == ss\*2) or (num == ss\*2+1)   def happy\_birthday(age):  a=1  while True:  if(fastCheck(age, a)): return a  a+=1 |

Тестирование:

|  |  |  |  |
| --- | --- | --- | --- |
| № | Тест | Ожидаемое значение | Полученное значение |
| 1 | 32 | 16 | 16 |
| 2 | 39 | 19 | 19 |
| 3 | 65 | 32 | 32 |
| 4 | 83 | 41 | 41 |
| 5 | 22 | 11 | 11 |

**Задача 4**

Условие задачи: Написать функцию tic\_tac\_toe(board), которая получает состояние поля для игры в крестики-нолики и опредляет победителя. Поле представлено в виде спсика 3x3, где значение равно 0 - точка пуста, 1 - это «X», или 2 - это «O». Функция должна возвращать: -1 если это ничья.

Решение задачи:

|  |
| --- |
| def tic\_tac\_toe(board):  def check(c):  def g(n):  return board[n // 3][n % 3]   if g(c[0]) == g(c[1]) and g(c[1]) == g(c[2]): return g(c[0])  return 0   combs = [  [0, 1, 2],  [3, 4, 5],  [6, 7, 8],  [0, 3, 6],  [1, 4, 7],  [2, 5, 8],  [0, 4, 8],  [2, 4, 6],  ];   for i in combs:  if check(i): return check(i)   for i in board:  for i2 in i:  if i2 == 0: return -1   return 0 |

Тестирование:

|  |  |  |  |
| --- | --- | --- | --- |
| № | Тест | Ожидаемое значение | Полученное значение |
| 1 | [0, 0, 1],  [0, 1, 2],  [2, 1, 0] | -1 | -1 |
| 2 | [1, 1, 1],  [0, 2, 2],  [0, 0, 0] | 1 | 1 |
| 3 | [1, 2, 1],  [1, 2, 2],  [2, 2, 1] | 2 | 2 |
| 4 | [2, 1, 2],  [2, 1, 1],  [1, 2, 1] | 0 | 0 |

**Задача 5**

Условие задачи: Создать список (библиотека книг), состоящий из словарей (книги). Словари должны содержать как минимум 5 полей (например, номер, название, год издания...). В список добавить хотя бы 10 словарей.

Решение задачи:

|  |
| --- |
| def info():  print('------------------------------------------------------')  for i in library:  print(f"[{i['id']}]: {i['author']} - {i['title']} ({i['age']}г.)")  print('------------------------------------------------------')   def getBookById():  id = int(input('Введите id книги: '))  for i in library:  if(i['id'] == id):  print(f"[{i['id']}]: {i['author']} - {i['title']} ({i['age']}г.)")  return  print(f'Не найдено книги по id [{id}]')   def olderThanCount():  y = int(input('Год: '))  a = 0  for i in library:  if(i['age'] < y): a+=1  print(f'Найдено {a} книг, год которых старше {y}')   def updateBook():  id = int(input('id корректируемой книги: '))  for i in library:  if(i['id'] == id):  i['author'] = input('Новый автор книги: ')  i['title'] = input('Новое название книги: ')  i['age'] = int(input('Новый год книги: '))  return  print(f'Не найдено книги по id [{id}]')   def deleteBook():  id = int(input('id удаляемой книги: '))  for i in library:  if (i['id'] == id):  library.remove(i)  return  print(f'Не найдено книги по id [{id}]') |

**2) Тема «Строки, работа с файлами»**

**Задача 1**

Условие задачи: Написать функцию break\_camel\_case, которая разбивает слова написанные CamelCase, используя в качестве разделителя пробел

Решение задачи:

|  |
| --- |
| def break\_camel\_case(s):  if sum(map(str.isupper, s)) < 2: return s;  for i in range(1, len(s)):  if s[i].isupper() and s[i-1] != ' ':  s = s[:i]+' '+s[i:]  return s |

Тестирование:

|  |  |  |  |
| --- | --- | --- | --- |
| № | Тест | Ожидаемое значение | Полученное значение |
| 1 | BreakCamelCase | Break Camel Case | Break Camel Case |
| 2 | helloWorld | helloWorld | helloWorld |
| 3 | helloWorld BreakCamelCase | helloWorld BreakCamelCase | helloWorld BreakCamelCase |

**Задача 2**

Условие задачи: Написать функцию blocks, которая получает строку, состоящую из букв и цифр и возвращает строку в виде блоков, разделенных символом дефис. Элементы блока должны быть отсортированы по принципу, указанному ниже, и каждый блок не может содержать несколько экземпляров одного и того же символа.

Решение задачи:

|  |
| --- |
| def sFunc(n):  if n.isdigit(): return 30 \* ord(n)  if n.isupper(): return 10 \* ord(n)  return ord(n)   def blocks(s):  s = sorted(s, key=sFunc)  cur\_block = []  ost = []  for i in s:  if i not in cur\_block: cur\_block.append(i)  else: ost.append(i)  return f"{''.join(cur\_block)}-{blocks(''.join(ost))}" if ost else ''.join(cur\_block) |

Тестирование:

|  |  |  |  |
| --- | --- | --- | --- |
| № | Тест | Ожидаемое значение | Полученное значение |
| 1 | 21AxBz | xzAB12 | xzAB12 |
| 2 | abacad | abcd-a-a | abcd-a-a |
| 3 | heyitssampletestkk | aehiklmpsty-ekst-est | aehiklmpsty-ekst-est |
| 4 | 6zjX9qcwTIuYNvdmL3CtElHa2n0rogKsSVPRWG4QAMUOe8JkyfxZDiBpb1Fh75GUTLMcbio7HO6rvn1NtDRmPJAejuXVFgaZI3pK90s4fBzqwEd5yWCQh8Sl2kxY | abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789-abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789 | abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789-abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789 |

**Задача 3**

Условие задачи: Дан список строк, состоящий из названий файлов с расширением. Написать функцию extension, которая возвращает список расширений наибольшего количества файлов

Решение задачи:

|  |
| --- |
| def extension(files):  a = {}  for i in files:  ext = i.split('.')[1]  try:  a[ext] += 1  except:  a[ext] = 1   a = sorted(a.items(), key=lambda item: item[1], reverse=True)  awns = []  for i in a:  if(i[1] == a[0][1]):  awns.append(f".{i[0]}")   return sorted(awns) |

Тестирование:

|  |  |  |  |
| --- | --- | --- | --- |
| № | Тест | Ожидаемое значение | Полученное значение |
| 1 | ["direful.pr", "festive.html", "historical.wav", "holistic.mp3", "impossible.jar", "gentle.cpp", "gleaming.xml", "inconclusive.js", "rect.jar", "befitting.mp3", "brief.wp", "beautiful.jar", "energetic.pt", "careful.wp", "defective.cpp", "icky.wav", "gorgeous.txt", "good.pt", "fat.pt", "bored.als", "adaptable.cpp", "fumbling.exe", "grieving.wp", "efficient.wav", "fearful.xml", "damp.html", "erect.exe", "annoyed.xml", "elderly.ala", "far-flung.txt", "careful.mp3", "actually.pt", "cynical.ala", "complex.exe", "extra-small.pt", "enchanted.ala", "amazing.html", "bashful.h", "hallowed.html", "entertaining.html", "bad.js", "illegal.maya", "deadpan.html", "furtive.wp", "hanging.css", "drunk.py", "capricious.wav", "damaging.Ue4", "cool.Ue4", "ambitious.css", "fortunate.wp", "electric.mp3", "crowded.txt", "cooperative.html", "graceful.pt", "aboard.pt", "exclusive.als", "glossy.css", "fluffy.pt", "cluttered.txt", "halting.cpp", "glib.cpp", "aback.pr", "cynical.Ue4", "chilly.xml", "hideous.ala", "finicky.txt", "feigned.ala", "better.Ue4", "dear.py", "available.xml", "easy.pr", "fine.mp3", "cowardly.jar", "incredible.css", "adhesive.exe", "energetic.mp3", "harmonious.exe", "general.als", "condemned.als", "flawless.als", "curvy.h", "ambitious.mp3", "disillusioned.xml", "bitter.h", "hanging.wp", "certain.cpp", "flashy.html", "cuddly.pr", "cagey.Ue4", "extra-small.pr", "amuck.cpp", "direful.html", "delightful.als", "helpless.h", "foamy.mp3", "enthusiastic.maya", "good.maya", "adhesive.css", "imperfect.pr", "bent.cpp", "exultant.zbrush", "adorable.mp3", "clammy.maya", "gaudy.pt", "blushing.css", "cuddly.Ue4", "curved.py", "boring.html", "broken.txt", "daily.jar", "giddy.xml", "curved.css", "future.maya", "graceful.css", "guiltless.maya", "gentle.cpp", "few.css", "calculating.txt", "clear.pr", "grey.py", "entertaining.ala", "elfin.txt", "excited.js", "abject.zbrush", "best.js", "boundless.wav", "hurried.ala", "delirious.cpp"] | [".cpp", ".html"] | [".cpp", ".html"] |
| 2 | ["dramatic.txt", "incompetent.jar", "alcoholic.wp", "clumsy.py", "abject.h", "boring.exe",  "aloof.pr", "familiar.py", "fanatical.py", "ill-informed.html", "fierce.pr", "accurate.html",  "grotesque.pr", "brown.py", "courageous.pt", "grouchy.jar", "giant.pt", "dirty.h",  "abaft.jar", "enormous.zbrush", "creepy.cpp", "beneficial.py", "absorbing.ala",  "heartbreaking.html", "exclusive.js", "fluttering.html", "happy.als", "fresh.pr",  "adamant.txt", "awful.maya", "frightening.maya", "bizarre.html", "efficacious.exe",  "illegal.wav", "dizzy.js", "gusty.wp", "delightful.pt", "full.als", "chivalrous.xml",  "filthy.js", "functional.jar", "conscious.wav", "feeble.exe", "hilarious.cpp", "earthy.py",  "handy.txt", "hollow.cpp", "aggressive.js", "fat.h", "drunk.exe", "clear.h", "easy.wav",  "eatable.pt", "grumpy.css", "empty.exe", "brief.jar", "aggressive.txt", "aggressive.txt",  "gruesome.ala", "awake.txt", "apathetic.mp3", "holistic.pt", "embarrassed.css", "flashy.maya",  "exultant.ala", "exuberant.exe", "graceful.pt", "dependent.py", "gigantic.wp", "husky.js",  "immense.pr", "defiant.cpp", "cooperative.html", "frantic.maya", "abashed.css",  "dysfunctional.h", "gusty.js", "dynamic.txt", "dreary.pt", "giddy.ala", "exciting.css",  "best.als", "humdrum.css", "busy.jar", "frail.cpp", "cagey.wav"] | [".pt", ".py", ".txt"] | [".pt", ".py", ".txt"] |
| 3 | ["crazy.pr", "black-and-white.als", "illegal.wav", "exultant.mp3", "exotic.jar", "capricious.pt",  "abundant.ala", "eatable.zbrush", "careful.py", "godly.css", "clever.txt", "dusty.maya",  "awesome.zbrush", "discreet.jar", "creepy.h", "fair.pt", "descriptive.mp3", "boundless.ala",  "berserk.xml", "hungry.exe", "awful.exe"] | [".ala", ".exe", ".jar", ".mp3", ".pt", ".zbrush"] | [".ala", ".exe", ".jar", ".mp3", ".pt", ".zbrush"] |

**Задача 4**

Условие задачи: Создать txt-файл, вставить туда любую англоязычную статью из Википедии.

Реализовать одну функцию, которая выполняет следующие операции:

- прочитать файл построчно;

- непустые строки добавить в список;

- удалить из каждой строки все цифры, знаки препинания, скобки, кавычки и т.д. (остаются латинские буквы и пробелы);

- объединить все строки из списка в одну, используя метод join и пробел, как разделитель;

- создать словарь вида {“слово”: количество, “слово”: количество, … } для подсчета количества разных слов,

где ключом будет уникальное слово, а значением - количество;

- вывести в порядке убывания 10 наиболее популярных слов, используя форматирование

(вывод примерно следующего вида: “ 1 place --- sun --- 15 times \n....”);

- заменить все эти слова в строке на слово “PYTHON”;

- создать новый txt-файл;

- записать строку в файл, разбивая на строки, при этом на каждой строке записывать не более 100 символов

при этом не делить слова.

Решение задачи:

|  |
| --- |
| def symbCheck(s):  s = ord(s)  if(  s == 32 or # пробел  97 <= s <= 122 or # a-z  65 <= s <= 90 # A-Z  ):  return True  return False  def replaceAll(text, dic):  for i, j in dic.items():  text = text.replace(i, j)  return text  def wiki\_function():  with open('wikiInput.txt') as file:  arr = []  for i in file.readlines():  if(i): arr.append(i)   clear\_arr = []  for i in arr:  str = ''  for i2 in i:  if symbCheck(i2): str+=i2  clear\_arr.append(str)   s = ' '.join(clear\_arr)  s = re.sub(' +', ' ', s)   dct = {}  for i in s.split(' '):  try:  dct[i] += 1  except:  dct[i] = 1   dct = sorted(dct.items(), key=lambda item: item[1], reverse=True)  tw = []  for i in range(10):  print(f"[{i+1}]: {dct[i][0]} {(15-len(dct[i][0])) \* ' '} {' ' if i!=9 else ''} | {dct[i][1]}")  tw.append(dct[i][0])   arr = []  for i in s.split(' '):  if i in tw: arr.append('PYTHON')  else: arr.append(i)   s = ''  t = 0  for i in arr:  if(t + len(i) > 100):  s+='\n'  t = 0  s += i + ' '  t += (len(i)+1)   with open('result.txt', 'w') as wr:  wr.write(s) |

**3) Тема «Основы объектно-ориентированного программирования»**

Класс Discipline.

Поля:  
 название: str

семестр: int

кафедра: str

Класс Academic. Производный от Discipline.

Поля:

название: str

семестр: int

кафедра: str

преподаватель: str

форма\_контроля: control\_form

часы: dict

Методы:

изменить\_преподавателя(str) : none,

изменить\_форму\_контроля(str) : none,

форматированная\_печать\_занятий() : str,

\_\_str\_\_() : str

Класс Practice. Производный от Discipline.

Поля:

название: str

семестр: int

кафедра: str

вид\_практики: practice\_type

руководитель\_практики: str

тематика\_практики : List

Методы:

изменить\_руководителя(str): none

добавить\_тематику(str): none

удалить\_тематику(str): none

изменить\_тематику(str, str) : none

\_\_str\_\_(): str

Класс Plan.

Поля:

код\_направления: str

название\_направления: str

кафедра: str

список\_основных\_дисциплин: List[Academic]

список\_практик: List[Practice]

Методы:

\_\_str\_\_(): str

\_\_len\_\_(): int

concatPract(): List

получить\_дисциплину(int): Discipline

изменить\_дисциплину(int): none

удалить\_дисциплину(int):none

\_\_add\_\_(Academic||Practice): None

WriteFile(): none

**Заключение**

В результате прохождения ознакомительной практики я освоил язык python, выполнил три практические работы, где понял основы ООП, в частности наследования на примере python, решил сложные прикладные задачи на строковые типы, массивы. Получил необходимые знания про обработку исключений и их вызов на python, про потоки ввода-вывода и чтение файлов, типизацию и многое другое.