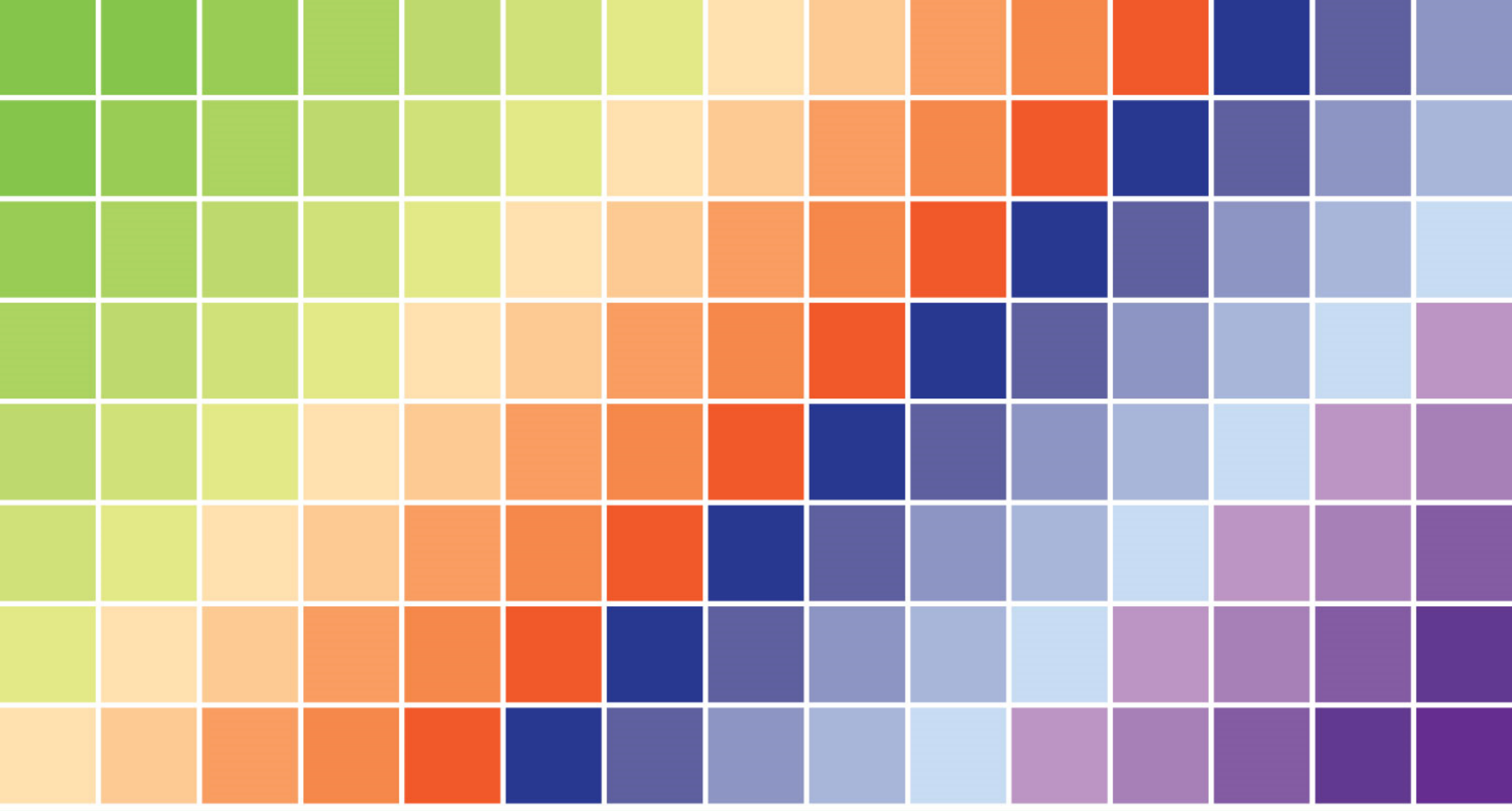


School of Computer Science

МОДЕЛЬ ПАМЯТИ В PYTHON. УПРАВЛЯЮЩИЕ КОНСТРУКЦИИ IF И WHILE

ПРОГРАММИРОВАНИЕ НА PYTHON

Лекции для IT-школы

ВОПРОСЫ ПО ПРОШЛОМУ ЗАНЯТИЮ

1. Как узнать список имен, определенных в текущей области видимости Python?
2. Как узнать список встроенных имен?
3. Как получить список ключевых слов?
4. В чем разница между ключевыми словами и встроенными именами?
5. Как удалить имя в локальном контексте?

ВОПРОСЫ ПО ПРОШЛОМУ ЗАНЯТИЮ

6. Какие кавычки используются для строк в Python? С какой целью?

7. Как размножить содержимое строки N раз?

8. С какой позиции индексируются строки?

9. Как обратиться к символу в строке с заданной позицией?

10. Как получить срез (часть) строки с заданной начальной и конечной позициями?

ВОПРОСЫ ПО ПРОШЛОМУ ЗАНЯТИЮ

11. Какого типа это выражение: 1.3356865e-3?

12. Каких операций с вещественными числами следует избегать для типа данных с плавающей точкой? Почему?

13. Как узнать минимально возможное значение с плавающей точкой в Python?
14. Как его можно использовать?

15. Какой тип данных в Python обеспечивает точную десятичную арифметику?

ВОПРОСЫ ПО ПРОШЛОМУ ЗАНЯТИЮ

Что такое исключение в программировании:

1. Исключительно хороший код, которому можно только позавидовать
2. Механизм обработки ошибочных ситуаций
3. Ошибка, при возникновении которой программа перестает работать

ВОПРОСЫ ПО ПРОШЛОМУ ЗАНЯТИЮ

– Какие из этих слов применяются при обработке исключений в Python?

1) disaster
2) try
3) problem
4) catch
5) except
6) call_police

ВОПРОСЫ ПО ПРОШЛОМУ ЗАНЯТИЮ

Как называется символьный тип в Python:

1. char
2. varchar
3. str
4. никак не называется

ВОПРОСЫ ПО ПРОШЛЫМ ЗАНЯТИЯМ

Укажите выражения, значения которых равны True:

1. '239' < '30' and 239 < 30
2. '239' < '30' and 239 > 30
3. '239' > '30' and 239 < 30
4. '239' > '30' and 239 > 30

МОДЕЛЬ ПАМЯТИ В PYTHON

Пространство имён, namespace:

Пространство объектов, heap:

Имя	Ссылка/Адрес	Значение	Тип
name	1237856	‘Вася’	str
amount	1237868	5678.78	float
b_25	1243464	4346	int
is_cold	4573735	True	bool
…	…	…	…

id(<переменная>) – получение адреса переменной в пространстве объектов

ОПЕРАТОРЫ ПРИСВАИВАНИЯ

1. Обычная ссылка на значение: var_1 = 1

2. Присваивание одного значения нескольким переменным: var_1 = var_2 = var_3 = 8

3. Множественное присваивание:
   var_1, var_2, var_3 = 5, 3.15, “Вася”

4. Присваивание, совмещенное с арифметической операцией:
   +=, -=, /=, //=, %=, *=

КОНСТАНТЫ

В Python нет ключевых слов для описания констант

По общепринятому соглашению константы в Python определяются в верхнем регистре

Пример:

```
>>> HOURS_IN_DAY = 24  # количество часов в сутках
>>>
>>> MINUTES_IN_HOUR  = 60  # количество минут в часе
>>>
>>> SECONDS_IN_MINUTE = 60   # количество секунд в минуте
>>>
>>> SECONDS_IN_HOUR = 3600  # количество секунд в часе
```

НЕИЗМЕНЯЕМОСТЬ ПРОСТЫХ ТИПОВ ДАННЫХ

- Типы данных в Python бывают изменяемые (mutable) и неизменяемые (immutable)
- Простые типы int, float, bool, str относятся к НЕизменяемым типам данных
- Значения неизменяемых типов данных не изменяются по ссылке
- Присваивая новое значение переменной неизменяемого типа мы получаем ссылку на НОВЫЙ объект этого типа

УСЛОВНЫЙ ОПЕРАТОР

Отступы обяза-тельны!

```
if <логическое выражение 1>:
    код, выполняемый при True
    для 1-го логического
    выражения…
[elif <логическое выражение 2>:
    код, выполняемый при True
    для 2-го логического
    выражения…
…]
[else:
    код, выполняемый, если все
    условия выше дают False
]
```

ВАРИАНТЫ ВЕТВЛЕНИЙ

Выражение	Описание
if <условие>: <блок>	Если <условие> истинно, то <блок> выполняется, иначе – пропускается

if <условие>:
  <блок 1>
else:
  <блок 2>
	Условная конструкция с условием else. Если <условие> истинно, выполняется <блок1>, если ложно – <блок2>

if <условие 1>:
  <блок 1>
elif <условие 2>:
  <блок 2>
…
elif <условие N>:
  <блок N>
else:
  <блок N+1>
	Условная конструкция с дополнительными условиями elif и необязательным else в конце. Будет исполнен единственный блок if / elif после первого условия, которое окажется истинным. Если же нет условий принимающих значение True, то будет исполнен блок после заключительного else

ОПЕРАТОР ВЫБОРА

```
блок-True if (условие) else блок-False
```

аналог оператора ? : в C++ и Java

Примеры:

```
max_var = var1 if (var1 > var2) else var2
```

```
print("Even" if (var_int % 2 == 0) else "Odd")
```

УСЛОВНЫЙ ОПЕРАТОР С IF. ПРИМЕР

```
pwd = input("Введите ваш пароль: ")

if pwd == "secret":
    print("Доступ предоставлен")
    <действия после предоставления
     доступа>
```

УСЛОВНЫЙ ОПЕРАТОР С IF-ELSE. ПРИМЕР

```
pwd = input("Введите ваш пароль: ")

if pwd == "secret":
    print("Доступ предоставлен")
    <действия после предоставления
     доступа>
else:
    print("Доступ запрещен")
    <действия после запрещения
     доступа>
```

УСЛОВНЫЙ ОПЕРАТОР С ELIF. ПРИМЕР

```
weight = int(input('Введите ваш вес:'))

if weight > 100:
    print('Вам пора в спортзал!')
elif weight < 50:
    print('Вам надо лучше питаться')
else:
    print('У вас нормальный вес')
```

ПРАКТИЧЕСКОЕ ЗАДАНИЕ №1. IF-ELSE

См. dice_guess_template.py

Написать программу, которая запрашивает целое число от 1 до 6.

Если введено число больше 6 или меньше 1, то программа должна вывести текст «Вы ввели неверное число».

Если число совпало с тем, что на кубике, то выводится текст «Вы угадали!»,
иначе выводится текст «Вы не угадали. На кубике выпало X»

СОВЕТЫ ПО ИСПОЛЬЗОВАНИЮ IF-ELIF-ELSE

- В сложных ветвлениях используйте ветку else даже если она вам сейчас не требуется:
  - Чтобы определить когда «что-то пошло не так»
  - Пример в mood_detector.py
- if-elif-elif не то же самое, что последовательность операторов if-if-if:
  - Среди **if-elif-elif**… выполнится **только один блок**
  - Пример в structured_if.py
- Вложенный if можно убрать в пользу внешнего if с объединением условий по and:
  - Пример в structured_if_with_and.py

Переменные grade1 и grade2 представляют баллы за 2 курса. Переменная num_passed равна нулю.
После каких фрагментов кода num_passed будет содержать количество пройденных экзаменов, по которым достигнут или превышен проходной балл (50 или более)?

1)

```
if grade1 >= 50:
    num_passed = num_passed + 1
if grade2 >= 50:
    num_passed = num_passed + 1
```

2)

```
if grade1 >= 50:
    num_passed = num_passed + 1
elif grade2 >= 50:
    num_passed = num_passed + 1
```

3)

```
if grade1 >= 50 and grade2 >= 50:
    num_passed = 2
if grade1 >= 50:
    num_passed = 1
if grade2 >= 50:
    num_passed = 1
```

4)

```
if grade1 >= 50 and grade2 >= 50:
    num_passed = 2
elif grade1 >= 50:
    num_passed = 1
elif grade2 >= 50:
    num_passed = 1
```

Рассмотрите этот блок кода:

```
if temperature > 28:
    if money >= 30:
        print("Я покупаю мороженку")
    else:
        print("Где можно найти тенёк?")
```

1)

```
if temperature > 28 and money >= 30:
    print("Я покупаю мороженку")
elif temperature > 28 and money < 30:
    print("Где можно найти тенёк?")
```

2)

```
if temperature > 28 and money >= 30:
    print("Я покупаю мороженку")
elif temperature > 28:
    print("Где можно найти тенёк?")
```

3)

```
if temperature > 28 and money >= 30:
    print("Я покупаю мороженку")
else:
    print("Где можно найти тенёк?")
```

Какие из приведенных кусков кода 1,2,3 ему идентичны?

Какое из решений выглядит наиболее элегантно?

Какой(ие) вариант(ы) отбросили и почему?

ПРАКТИЧЕСКОЕ ЗАДАНИЕ №2. РАСЧЕТ ШТРАФА

- На участке дороги установлено ограничение скорости в 40 км/ч
- При скорости 140 км/ч и выше водитель лишается прав
- Если скорость в 40 км/ч превышена то за каждые лишние 20 км/ч к штрафу доначисляется 500 штрафных рублей и определяется общая сумма штрафа
- Программа должна принимать на вход скорость автомобиля в км/ч и выдавать сумму штрафа, либо сообщения «Штрафа нет» / «Лишение прав!»

ЦИКЛ WHILE

Отступы обяза-тельны!

```
while <логическое выражение>:
    код, выполняемый при True
    для логического
    выражения…
[else:
    код, выполняемый, если блок
    while нормально завершился
    или не выполнялся вовсе
]
```

КАК МЫ БЫ ЖИЛИ БЕЗ ЦИКЛОВ…

```
>>> figure = 7
>>> multiplier = 1
>>> print(figure, 'x', multiplier, '=', figure * multiplier)
7 x 1 = 7
>>> multiplier = multiplier + 1
>>> print(figure, 'x', multiplier, '=', figure * multiplier)
7 x 2 = 14
>>> multiplier = multiplier + 1
>>> print(figure, 'x', multiplier, '=', figure * multiplier)
7 x 3 = 21
>>> multiplier = multiplier + 1
>>> print(figure, 'x', multiplier, '=', figure * multiplier)
7 x 4 = 28
>>> multiplier = multiplier + 1
>>> print(figure, 'x', multiplier, '=', figure * multiplier)
7 x 5 = 35
>>> multiplier = multiplier + 1
>>> print(figure, 'x', multiplier, '=', figure * multiplier)
7 x 6 = 42
>>> multiplier = multiplier + 1
>>> print(figure, 'x', multiplier, '=', figure * multiplier)
7 x 7 = 49
>>> multiplier = multiplier + 1
>>> print(figure, 'x', multiplier, '=', figure * multiplier)
7 x 8 = 56
>>> multiplier = multiplier + 1
>>> print(figure, 'x', multiplier, '=', figure * multiplier)
7 x 9 = 63
```

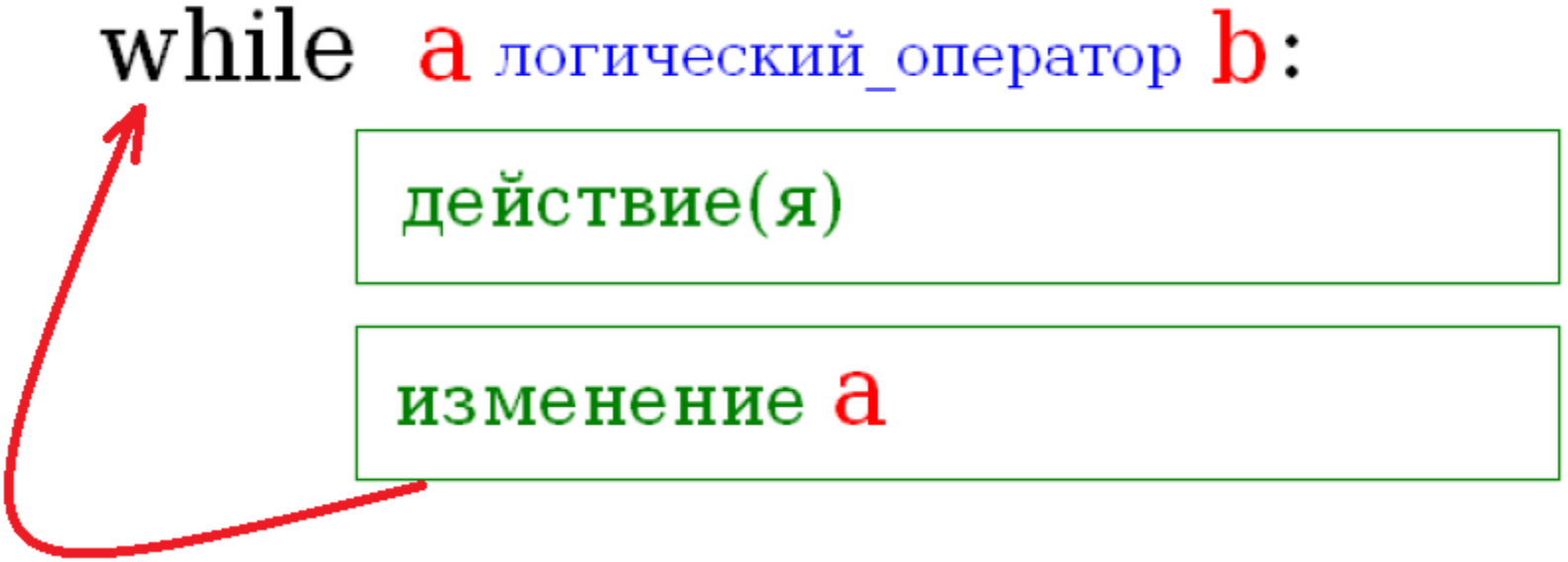
ЦИКЛ ОБЛЕГЧАЕТ ПОВТОРЕНИЕ ОДНОТИПНЫХ ДЕЙСТВИЙ

«Быстрая» таблица умножения для семерки:

```
>>> figure = 7
>>> multiplier = 1
>>> while multiplier < 10:
        print(figure, 'x', multiplier, '=', figure * multiplier)
        multiplier = multiplier + 1


7 x 1 = 7
7 x 2 = 14
7 x 3 = 21
7 x 4 = 28
7 x 5 = 35
7 x 6 = 42
7 x 7 = 49
7 x 8 = 56
7 x 9 = 63
>>>
```

ЦИКЛ WHILE. ТИПОВАЯ СТРУКТУРА



Изменение условия цикла в его теле **обязательно**, если не используется **break**

ЦИКЛ WHILE. ОПРЕДЕЛЕНИЯ

- Блок кода, исполняемый в цикле, называется ***итерацией*** или ***телом цикла***
- Обычно перед входом в цикл инициируют ***управляющую переменную***, а внутри цикла **while** ее значение **изменяют**
- Даже при изменении управляющей переменной в теле цикла нужно следить, чтобы условие цикла хоть когда-нибудь становилось ложным

ПРАКТИЧЕСКОЕ ЗАНЯТИЕ НА WHILE

См. losing_battle_bad.py

- Программа, моделирует последнюю битву главного героя в Action-игре
- Если здоровья биться с троллями уже не хватает, то цикл битвы должен завершиться
- Так в чем же проблема?

НАМЕРЕННО «БЕСКОНЕЧНЫЙ» ЦИКЛ

- Рассмотрите скрипт strange_counter.py
- Команда break внутри тела цикла передает управление за его пределы
- Команда continue внутри тела цикла передает управление на его очередную итерацию:
  - То есть происходит возврат к началу цикла
  - При этом снова проверяется условие цикла и, если оно True, то тело цикло снова будет выполняться

ВЛОЖЕННЫЕ ЦИКЛЫ WHILE. ПРИМЕР С ELSE

Поиск простых чисел до NUM_MAX:

```
>>> NUM_MAX = 20                          # верхняя граница диапазона для поиска простых чисел (константа)
>>> num = 2                               # проверяемое число будет лежать в диапазоне от num до NUM_MAX
>>> while num < NUM_MAX:                  # цикл для перебора num
        div = 2                           # проверим будет ли num делиться на div, начиная с div = 2
        while div < num:
            if num % div == 0:            # num делится на div - это число имеет целочисленный делитель
                break                     # т.е. оно составное, перейдем к следующему
            div += 1                      # это аналог присваивания div = div + 1
        else:
            print(num, "простое число") # не нашлии целочисленный делитель в цикле - это простое число
        # конец внутреннего цикла while div < num
        num += 1                          # это аналог присваивания num = num + 1
    # конец внешнего цикла while num < NUM_MAX


2 простое число
3 простое число
5 простое число
7 простое число
11 простое число
13 простое число
17 простое число
19 простое число
>>>
```

Смотрите скрипт prime_numbers.py

ПРАКТИЧЕСКОЕ ЗАДАНИЕ №3. ТАБЛИЦА УМНОЖЕНИЯ ДО 10-ТИ

- Второклассник учит таблицу умножения
- Сегодня он должен сдать её наизусть, но вот беда, забыл некоторые результаты
- Напишите программу, которая распечатает шпаргалку малышу:

```
 1   2   3   4   5   6   7   8   9  10
 2   4   6   8  10  12  14  16  18  20
 3   6   9  12  15  18  21  24  27  30
 4   8  12  16  20  24  28  32  36  40
 5  10  15  20  25  30  35  40  45  50
 6  12  18  24  30  36  42  48  54  60
 7  14  21  28  35  42  49  56  63  70
 8  16  24  32  40  48  56  64  72  80
 9  18  27  36  45  54  63  72  81  90
10  20  30  40  50  60  70  80  90 100
```

ПЛАНИРОВАНИЕ НА ПСЕВДОКОДЕ

- Алгоритм зарабатывания $ 1 000 000:
  - Если вы способны придумать новый полезный товар – выпустите его в свет,
    иначе
  - Выпустите существующий товар под своей маркой

- А именно:
  - Создайте рекламный ролик своего товара
  - Покажите этот ролик по TV
  - Назначьте цену $100 за единицу товара
  - Продайте 10 000 единиц товара

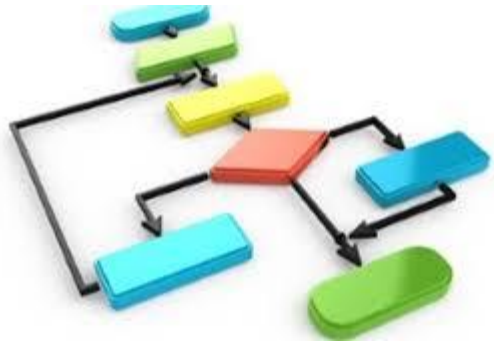
ПРАКТИЧЕСКОЕ ЗАДАНИЕ №4. ИГРА «ОТГАДАЙ ЧИСЛО»

- Программа загадывает случайное число в диапазоне от 1 до 100
- Пользователь пытается угадать это число
- Если не угадал, то выдается подсказка:
  - «Вы ввели меньшее число»
    или
  - «Вы ввели большее число»
- Цикл подбора числа продолжается, пока человек не угадает загаданное число

1-Я ВЕРСИЯ ПСЕВДОКОДА ПРОГРАММЫ «ОТГАДАЙ ЧИСЛО»

- Выбрать случайное число
- До тех пор, пока игрок его не отгадает:
  - Предоставить игроку возможность отгадывать
- Поздравить игрока

guess_num_template.py



2-Я ВЕРСИЯ ПСЕВДОКОДА ПРОГРАММЫ «ОТГАДАЙ ЧИСЛО»

- Поздороваться с игроком и объяснить ему правила игры
- Выбрать случайное число от 1 до 100
- Установить переменные для отгадывания и количества попыток
  - Пока указанное игроком число не совпадает с загаданным:
    - если оно больше загаданного – запросить число поменьше
    - иначе – запросить число побольше
  - Вновь предложить игроку отгадать число
  - Увеличить порядковый номер попытки на 1
- Поздравить игрока с победой
- Сообщить сколько попыток потребовалось

ДОМАШНЕЕ ЗАДАНИЕ. ИГРА «ОТГАДАЙ ЧИСЛО» V.2

- Измените программу таким образом, чтобы у игрока было ограниченное количество попыток:
  1) Указанное в константе в тексте программы
  2) Запрошенное у игрока на старте. Если игрок вводит пустое значение, то использовать ограничение по умолчанию из пункта 1
  3)* Указанное в качестве параметра командной строки, а если отсутствует, то см. пункт 2

- Если игрок не укладывается в заданное число и проигрывает то программа должна выводить очень суровый текст

ДОМАШНЕЕ ЗАДАНИЕ. ИГРА «ЗАДУМАЙ ЧИСЛО»

- Напишите на псевдокоде алгоритм игры, в которой число от 1 до 100 загадывает человек, а отгадывает компьютер
- Какой должна быть оптимальная стратегия отгадывания?
- Если алгоритм на псевдокоде будет удачным, реализуйте игру на Python

СПАСИБО ЗА ВНИМАНИЕ !

ВОПРОСЫ ?

School of Computer Science