Тема: Функции Python
Цель урока: знакомство учащихся с функцией в Python, понятием функции, а так же с командой создания функции def(); создание программы на языке Python с использованием функции.
Задачи урока:
Образовательные: познакомить учащихся с функцией, понятием функции, с инструкцией def(); формирование умений и навыков записи функции на языке программирования Python; определять цель работы; выбирать рациональные способы выполнения работы; получение новых знаний (знакомство с новыми понятиями).
Воспитательные: умение следовать принципам диалогического, субъ- ектно-субъектного общения; умение работать в классе развивать информационную культуру учащихся; способность к самостоятельной и коллективной деятельности.
Развивающие: развить навыки программирования в среде программирования Python; развить алгоритмическое мышление учащихся; развитие умственной деятельности (выполнения операций анализа).

Ход урока:

Усвоение нового материала.
Личностные УУД: осознание ответственности за общее дело; нравственно-этическое оценивание усваиваемого содержания.
Познавательные УУД: анализ, синтез, сравнение, обобщение; извлечение необходимой информации; подведение под понятие.
Коммуникативные УУД: выражение своих мыслей с достаточной полнотой и точностью; формулирование и аргументация своего мнения в коммуникации.
Регулятивные УУД: Контроль, коррекция, оценка; волевая саморегуляция в ситуации затруднения.
Функции в программировании можно представить как изолированный блок кода, обращение к которому в процессе выполнения программы может быть многократным. Зачем нужны такие блоки инструкций? В первую очередь, чтобы сократить объем исходного кода: рационально вынести часто повторяющиеся выражения в отдельный блок и, затем, по мере надобности, обращаться к нему.
Представим себе следующую ситуацию. Требуется написать скрипт, который при выполнении должен три раза запрашивать у пользователя разные данные, но выполнять с ними одни и те же действия.

```
a = int(input('Введите первое число: '))
b = int(input('Введите второе число: '))
if a > b:
    print(a-b)
else:
    print(b-a)

c = int(input('Введите первое число: '))
d = int(input('Введите второе число: '))
if c > d:
    print(c-d)
else:
    print(d-c)

e = int(input('Введите первое число: '))
f = int(input('Введите второе число: '))
if e > f:
    print(e-f)
else:
    print(f-e)
```

Данная программа находит модуль разницы двух чисел. Очевидно, что такая запись исходного кода не рациональна: получаются три почти одинаковых блока кода. Почему бы не использовать цикл while для организации повторения?

```
i = 0
while i < 3:
    a = int(input('Введите первое число: '))
    b = int(input('Введите второе число: '))
    if a > b:
        print(a-b)
    else:
        print(b-a)
    i = i + 1
```

Однако, в этом случае есть один нюанс. Вводимые пользователем данные всегда связываются с переменными a и b. При каждом витке цикла прежние данные утрачиваются. Что же делать, если все шесть чисел, введенных пользователем надо сохранить для дальнейшего использования в программе? Рассмотрим решение этой задачи с использованием функции.

```
def diff():
    m = int(input('Введите первое число: '))
    n = int(input('Введите второе число: '))
    if m > n:
        print(m-n)
    else:
        print(n-m)
    return m,n

a,b = diff()
c,d = diff()
e,f = diff()
```

def - это инструкция (команда) языка программирования Python, позволяющая создавать функцию. diff - это имя функции, которое (так же как и имена переменных) может быть почти любым, но желательно осмысленным. После в скобках перечисляются параметры функции. Если их нет, то скобки остаются пустыми. Далее идет двоеточие, обозначающее окончание заголовка функции (аналогично с условиями и циклами). После заголовка с новой строки и с отступом следуют выражения тела функции. В конце тела функции присутствует инструкция return (может и не быть), которая возвращает значение(я) в основную ветку программы. В данном случае, если бы в функции не было инструкции return, то в основную программу ничего бы не возвращалось, и переменным a и b (c и d, а также e и f) числовые значения не присваивались бы.

После функции идет, так называемая, основная ветка программы, в которой переменным попарно присваивается результат выполнения вызываемой функции. В иных ситуациях, когда функция не возвращает значений, ее вызов не связывается с переменной.

Выражения тела функции выполняются лишь тогда, когда она вызывается в основной ветке программы. Так, например, если функция присутствует в исходном коде, но нигде не вызывается в нем, то содержащиеся в ней инструкции не будут выполнены ни разу.

- **Первичное закрепление.**

Личностные УУД: нравственно-этическое оценивание усваиваемого содержания.

Познавательные УУД: анализ, синтез, сравнение, обобщение; извлечение необходимой информации.

Коммуникативные УУД: выражение своих мыслей с достаточной полнотой и точностью.

Регулятивные УУД: Контроль, коррекция, оценка.

Для закрепления полученных знаний предлагаю выполнить практическую работу № 11. «Функции в программировании».

Примерное выполнение практической работы:

1.

```
def troyka():
    a = 2
    b = 3
    c = 1
    value = (a + b + c)
    print (value)
    return value
troyka()
```

2.

```
def one(a,b):
        c=int(a)+int(b)
        print(c)
        two(c)
def two(c):
        d=str(c)
        for i in d:
                print('-',i,'-')
x=input('input number 1: ')
y=input('input number 2: ')
one(x,y)
```

- **Сообщение домашнего задания.**
  Личностные УУД: осознание ответственности за общее дело; нравственно-этическое оценивание усваиваемого содержания.
  Познавательные УУД: контроль и оценка процесса и результатов деятельности.
  Коммуникативные УУД: планирование учебного сотрудничества.
  Регулятивные УУД: Контроль, коррекция, оценка.

Задания:

https://taskcode.ru/function/average
https://younglinux.info/python/task/maths-function
Ответы отправлять на почту - magus-grand@mail.ru
Программа рассчитана на два урока 23.03.2020 -25.03.2020