Maktab Sharif

Python | Main course

Session 2

- Strings
- Lists
- Loops
- (Exercises)

by Mohammad Amin H.B. Tehrani

www.maktabsharif.ir

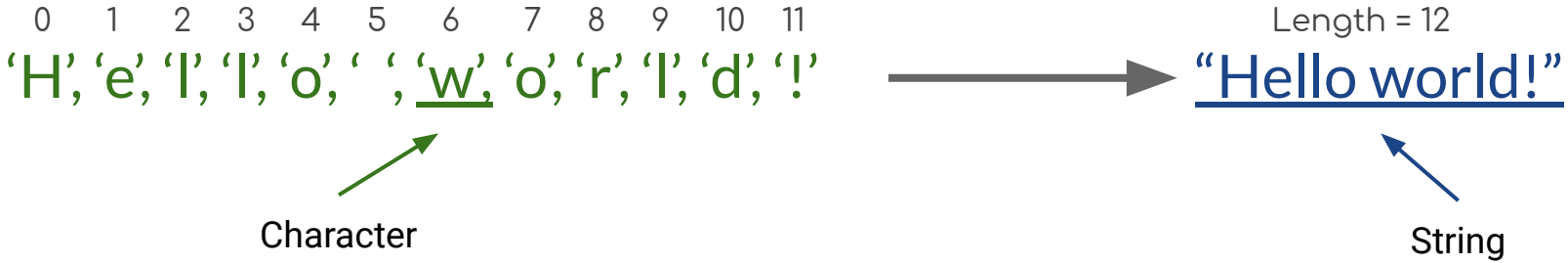
Chapter 6

Strings

Maktab Sharif

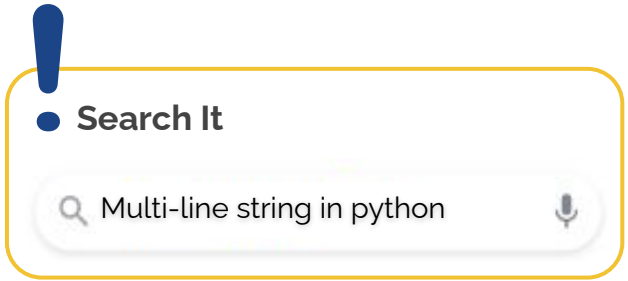
String

a **String** is traditionally a sequence of characters



Strings in python are surrounded by either single quotation marks, or double quotation marks.

"Hello world!" == 'Hello world!'



String operators

Maktab Sharif

In general, you cannot perform mathematical operations on strings, even if the strings look like numbers. The following are **illegal**:

```
message = "hello"
message - 1
"Hello" / 123
message * "world"
"15" + 2
"15" ** 2
```

```
message = "hello"
print(message + " " + "world!")    # It's OK
print(message * 3)                 # It's OK
```

String methods

Maktab Sharif

Strings are also objects. Each string instance has its own attributes and methods. The most important attribute of the string is the collection of characters. There are a wide variety of methods. Try the following program.

Some of str methods (functions): Read more

1.	.lower()	-> str	Converts a string into lower case
2.	.upper()	-> str	Converts a string into upper case
3.	.split()	-> list	Splits the string at the specified separator, and returns a list
4.	.islower()	-> bool	Returns True if all characters in the string are lower case
5.	.isupper()	-> bool	Returns True if all characters in the string are upper case
6.	.isnumeric()	-> bool	Returns True if all characters in the string are numeric
7.	.isalpha()	-> bool	Returns True if all the characters are alphabet letters (a-z)
8.	.count(s)	-> int	Returns the number of times a specified value occurs in a string
9.	.startswith(s)	-> bool	Returns true if the string starts with the specified value
10.	.endswith(s)	-> bool	Returns true if the string ends with the specified value

.lower() & .upper()

Maktab Sharif

The **upper()** method returns a string where all characters are in upper case.
The **lower()** method returns a string where all characters are lower case.

```
s = "Hello world!"

print(s.lower())
print(s.upper())

print(s[:5].lower())
print(s[6:].upper())

print(s[0].lower())
print(s[6].upper())
```

.split()

Maktab Sharif

The **split()** method splits a string into a list.

```
s = "Hey you, john. How Are you?"
print(s.split())
print(s.split(','))
print(s.split('.'))
print(s.split('john.'))

s = "1 23 1.2 11 0.5 -4 5 -5.5"
s_list = s.split()
print(sum(s_list))
int_list = list(map(float, s_list))
print(sum(int_list))
```

.islower() & .isupper() & .isnumeric() & .isalpha() & ...

Maktab Sharif

- The **islower**() method returns True if all the characters are in lower case, otherwise False.
- The **isupper**() method returns True if all the characters are in upper case, otherwise False.
- The **isnumeric**() method returns True if all the characters are numeric (0-9), otherwise False.
- The **isalpha**() method returns True if all the characters are alphabet letters (a-z).
- The **isalnum**() method returns True if all the characters are alphanumeric, meaning alphabet letter (a-z) and numbers (0-9).

```
print('akbar'.islower(), 'akBar'.islower())
print('REZA'.isupper(), 'REzA'.isupper())
print('12443'.isnumeric(), '-12/6'.isnumeric())
print('HidudeHowareyou'.isalpha(), 'Hi dude, How are you?'.isalpha())
print('Kobra11'.isalnum(), 'Kobra-11'.isalnum())
```

.count(str_value)

Maktab Sharif

The **count**() method returns the number of times a specified value appears in the string.

```
my_string = "Hello hello Hello hello Hello"

print(my_string.count(' '))
print(my_string.count('Hello'))
print(my_string.count('hello'))
print(my_string.count('ll'))
```

Str Slices

Maktab Sharif

Return special part of str (Substring) : ur_str[start_index : end_index]

```
string = "Hello world!!!"

print(string[1:3])       # ?
print(string[2:4])       # ??
print(string[0:2])       # ???
print(string[:2])        # ????
print(string[2:])        # ?????
print(string[:])         # ??????
print(string[-2:])       # ???????
print(string[:-2])       # ????????
```

Str Slices with Step

Maktab Sharif

Return Slice of string with **step**: ur_str[start_index : end_index : **step**]

```
string = "Hello world, bro!"

print(string[::2])
print(string[::3])
print(string[2:6:2])
print(string[:3:2])
print(string[3::3])
print(string[::-1])
print(string[::-2])
print(string[::2][:3])
print(string[::-1][4:])
```

String in & not in operators

Maktab Sharif

Check substring s **exists** in string: **s in string**

➔ Returns True if a substring **s** is present in the string.

Check substring s **NOT exists** in string: **s not in string**

➔ Returns True if a substring **s** is not present in the string.

```
print('p' in 'apple')
print('i' in 'apple')
print('ap' in 'apple')
print('pa' in 'apple')

print('a' in 'a')
print('apple' in 'apple')
print('' in 'a')
print('' in 'apple')
```

```
print('p' not in 'apple')
print('i' not in 'apple')
print('ap' not in 'apple')
print('pa' not in 'apple')

print('a' not in 'a')
print('apple' not in 'apple')
print('' not in 'a')
print('' not in 'apple')
```

Chapter 7

Lists

Maktab Sharif

Lists

Maktab Sharif

When we want to store Multiple value into one variable, We use lists.

A **list** is a sequential collection of Python data values, where each value is identified by an index. The values that make up a list are called its **elements**. Lists are similar to strings, which are ordered collections of characters, except that the elements of a list can have any type and for any one list, the items can be of different types.
There are several ways to create a new list. The simplest is to enclose the elements in square brackets ([and]).

```
numbers = [-2, 2, 3, 11]
cities = ['Tehran', 'Karaj', 'Qom']
my_list = [1, 'Yes', 11.222, True ]
akbar_info = ['Akbar', 'Rezaii', 22, 185.5, 'akbar@gmail.com']
empty_list = []
students_id = [87, 23, 123, 22, 48, 16]
```

Access to List Items

Maktab Sharif

Use index operator: ur_list[i]

The indexing operator (Python uses square brackets to enclose the index) selects a single element from a list.

We use the index operator ([] – not to be confused with an empty list). The expression inside the brackets specifies the index. Remember that the indices start at 0. Any integer expression can be used as an index and as with strings, negative index values will locate items from the right instead of from the left.

```
numbers = [-2, 2, 3, 11]

print(numbers[1])    # ?
print(numbers[2])    # ??
print(numbers[3])    # ???
print(numbers[0])    # ????
print(numbers[4])    # ?????
```

Access to List Items

Maktab Sharif

Use index operator: ur_list[i]

The indexing operator (Python uses square brackets to enclose the index) selects a single element from a list.

We use the index operator ([] – not to be confused with an empty list). The expression inside the brackets specifies the index. Remember that the indices start at 0. Any integer expression can be used as an index and as with strings, negative index values will locate items from the right instead of from the left.

```
numbers = [-2, 2, 3, 11]

print(numbers[1])    # 2
print(numbers[2])    # 3
print(numbers[3])    # 11
print(numbers[0])    # -2
print(numbers[4])    # ERROR: Index of Bound!
```

Note

List indexes Starts from **0**

Length of List

Maktab Sharif

Use function: len(ur_list)

As with strings, the function len returns the length of a list (the number of items in the list). However, since lists can have items which are themselves lists, it important to note that len only returns the top-most length. In other words, sublists are considered to be a single item when counting the length of the list.

```
numbers = [-2, 2, 3, 11]
cities = ['Tehran', 'Karaj', 'Qom']
my_list = [1, 'Yes', 11.222, True ]
akbar_info = ['Akbar', 'Rezaii', 22, 185.5, 'akbar@gmail.com']

print(len(numbers))      # ?
print(len(cities))       # ??
print(len(my_list))      # ???
print(len(akbar_info))   # ????
```

Add (Append) item to List

Maktab Sharif

Append new Item to end of list: ur_list.append(ur_new_item)

The append method adds a new item to the end of a list. It is also possible to add a new item to the end of a list by using the concatenation operator. However, you need to be careful.

```
numbers = [-2, 2, 3, 11]
print(numbers)    # ?

numbers.append(12)
print(numbers)    # ??

numbers.append([12,13])
print(numbers)    # ???

numbers += [12,13]
print(numbers)    # ????
```

Insert item into List

Maktab Sharif

Insert new item into special position: ur_list.insert(index, item)

We have learned that we can put an element to the end of a list by using the method "append". To work efficiently with a list, we need also a way to add elements to arbitrary positions inside of a list. This can be done with the method "insert"

```
numbers = [-2, 2, 3, 11]
numbers.insert(1,4)
print(numbers[1], numbers)     # ?

numbers.insert(4, 1)
print(numbers[4], numbers)     # ??

numbers.insert(9,9)
print(numbers[-1], numbers)    # ???
```

Delete item from list (pop)

Maktab Sharif

Removes item at the special position : ur_list.pop(index)

The pop(index) method removes the element at the specified position (index) then returns the removed item.

pop() method (without index) removes the first element from the list. (index=0)

```
numbers = [-2, 2, 3, 11]
print(numbers.pop())          # ?
print(numbers)              # ??

print(numbers.pop(2))         # ???
print(numbers)              # ????

print(numbers.pop(2))         # ?????
print(numbers)              # ??????

print(numbers.pop(5))         # ???????
```

Item Membership in List

Maktab Sharif

Check item **exists** in list: **x in ur_list**

➔ Returns True if a sequence with the specified value is present in the object.

Check item **NOT exists** in list: **x not in ur_list**

➔ Returns True if a sequence with the specified value is not present in the object.

```
numbers = [-2, 2, 3, 11]

print(-2 in numbers)           # ?
print(-2 not in numbers)       # ??
print(0 in numbers)            # ???
print(11 not in numbers)       # ????
print("HI" in numbers)         # ?????
print(True not in numbers)     # ??????
print(True in numbers)         # ???????
```

List Slices

Maktab Sharif

Return special part of list: ur_list[start_index : end_index]

So every time you want to extract part of a string or a list, you use in Python the slice operator. The syntax is simple. Actually it looks a little bit like accessing a single element with an **index**, but instead of just one number we have more, separated with a colon ":". We have a start and an end **index**, one or both of them may be missing.

```
numbers = [-2, 2, 3, 11]

print(numbers[1:3])        # ?
print(numbers[2:4])        # ??
print(numbers[0:2])        # ???
print(numbers[:2])         # ????
print(numbers[2:])         # ?????
print(numbers[:])              # ??????
print(numbers[-2:])        # ???????
print(numbers[:-2])        # ????????
```

List Slices with Step

Maktab Sharif

Return Slice of list with step: ur_list[start_index : end_index : step]

Slicing works with three arguments as well. If the third argument is for example 3, only every third element of the list, string or tuple from the range of the first two arguments will be taken.

```
alphabets = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h']

print(alphabets[::2])          # ?
print(alphabets[::3])          # ??
print(alphabets[2:6:2])        # ???
print(alphabets[:3:2])         # ????
print(alphabets[3::3])         # ?????
print(alphabets[::-1])         # ??????
print(alphabets[::-2])         # ???????
print(alphabets[::2][:3])      # ????????
print(alphabets[::-1][4:])     # ?????????
```

An Example

Maktab Sharif

```
my_list = ['ali', 'akbar', 'reza', 'mamad', 'nosrat', 'saman']

print(len(my_list))                                          # ?
print(len(my_list[2:5:2]))                                   # ??
print(my_list[0],my_list[4],my_list[-1])                     # ???
print(my_list[2:][0],my_list[:4][-1],my_list[::-1][0])       # ????

x = my_list.pop(1)
print(x)                                                     # ?????
print(x in my_list)                                          # ??????

my_list.insert(4, x)
print('akbar' in my_list[:4])                                # ???????
print(my_list[:4])                                           # ????????

my_list.append(my_list.pop())
print('ali' not in my_list)                                  # ?????????
print(my_list[::-1])                                         # ??????????
```

An Example

Maktab Sharif

```
my_list = ['ali', 'akbar', 'reza', 'mamad', 'nosrat', 'saman']

print(len(my_list))                                    # 6
print(len(my_list[2:5:2]))                             # 2
print(my_list[0],my_list[4],my_list[-1])               # ali nosrat saman
print(my_list[2:][0],my_list[:4][-1],my_list[::-1][0]) # reza mamad saman

x = my_list.pop(1)
print(x)                                               # akbar
print(x in my_list)                                    # False

my_list.insert(4, x)
print('akbar' in my_list[:4])                          # False
print(my_list[:4])                                     # ['ali', 'reza', 'mamad', 'nosrat']

my_list.append(my_list.pop())
print('ali' not in my_list)                            # False
print(my_list[::-1])                # ['saman', 'akbar', 'nosrat', 'mamad', 'reza', 'ali']
```

Chapter 8

Loops (while, for)

Maktab Sharif

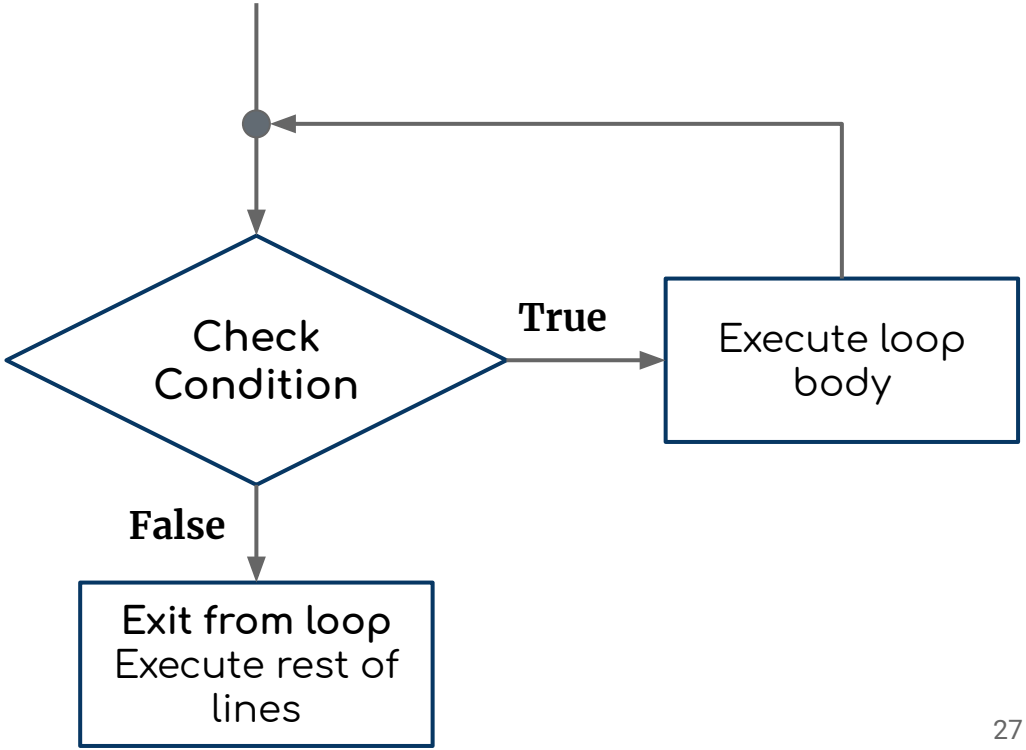
Loops

Maktab Sharif

Many algorithms make it necessary for a programming language to have a construct which makes it possible to carry out a sequence of statements repeatedly. The code within the loop, i.e. the code carried out repeatedly, is called the body of the loop.

```
while BOOL_EXPRESSION:
    STATEMENTS
    ...
    ...
```

Don't Forget TABs!



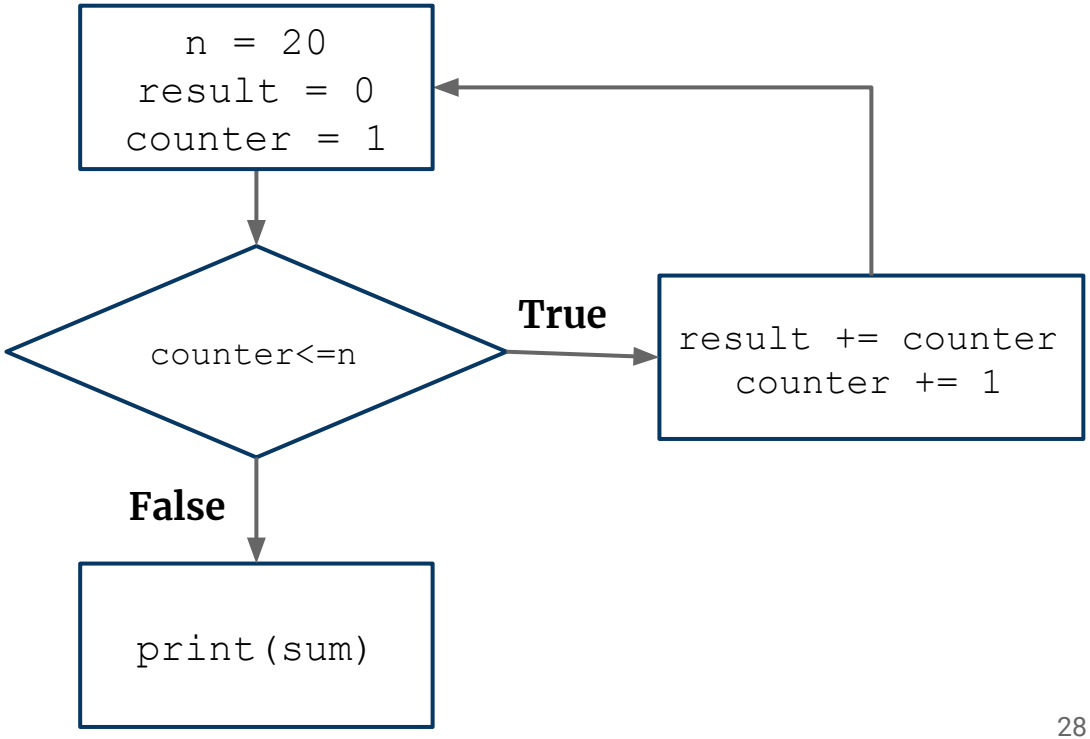
while loop

```
n = 20

result = 0
counter = 1

while counter <= n:
    result += counter
    counter += 1

print(result)          # ?
```

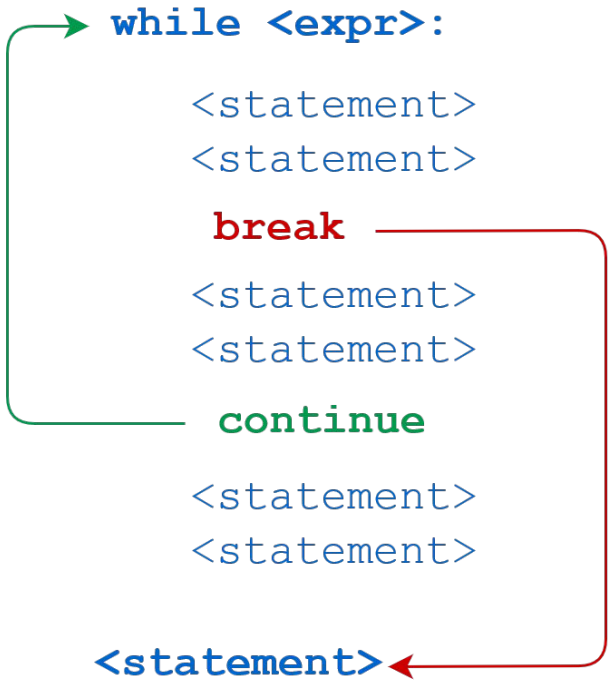


Break and Continue statements

Maktab Sharif

In each example you have seen so far, the entire body of the while loop is executed on each iteration. Python provides two keywords that terminate a loop iteration prematurely:

- The Python **break** statement immediately terminates a loop entirely. Program execution proceeds to the first statement following the loop body.
- The Python **continue** statement immediately terminates the current loop iteration. Execution jumps to the top of the loop, and the controlling expression is re-evaluated to determine whether the loop will execute again or terminate.



Break and Continue Example

Maktab Sharif

What's output of the code below

```
i = 0

while i < 30:
    i += 1
    print(i)

    if i % 3 == 0:
        i += 1
        continue
    if i//11:
        break
print("The end")
```

Break and Continue Example

Maktab Sharif

What's output of the code below

```
i = 0

while i < 30:
    i += 1
    print(i)

    if i % 3 == 0:
        i += 1
        continue
    if i//11:
        break
print("The end")
```

Output:

```
1
2
3
5
6
8
9
11
The end
```

For loops

Maktab Sharif

Like the while loop the for loop is a programming language statement, i.e. an iteration statement, which allows a code block to be repeated a certain number of times.

As we mentioned earlier, the Python for loop is an iterator based for loop. The Python for loop starts with the keyword "for" followed by an arbitrary variable name, which will hold the values of the following sequence object, which is stepped through. The general syntax looks like this:

```
for variable in your_list:
    STATEMENTS
    ...
    ...
```

Equal While statement:

```
i = 0
while i < len(your_list):
    variable = your_list[i]
    i += 1
    STATEMENTS
    ...
    ...
```

Example

Maktab Sharif

Say hello to present students:

Write a program, that says hello to all of students in the attendance list.

```
attendance_list = [
        'Akbar',
        'Reza',
        'Saman',
        'Mojtaba',
        'Kasra'
]
```

Example: using while loop

Maktab Sharif

Say hello to present students:

Write a program, that says hello to all of students in the attendance list.

```
attendance_list = [
        'Akbar',
        'Reza',
        'Saman',
        'Mojtaba',
        'Kasra'
]
```

Using While loop:

```
i = 0
while i < len(attendance_list):
    student = attendance_list[i]
    i += 1
    print("Hello", student)
```

Example: using while loop

Maktab Sharif

Say hello to present students:

Write a program, that says hello to all of students in the attendance list.

```
attendance_list = [
        'Akbar',
        'Reza',
        'Saman',
        'Mojtaba',
        'Kasra'
]
```

Using While loop:

```
i = 0
while i < len(attendance_list):
    student = attendance_list[i]
    i += 1
    print("Hello", student)
```

Using **for** loop:

```
for student in attendance_list:
    print("Hello", student)
```

range() function

Maktab Sharif

Generate a sequence of numbers: **range(...)**

- ➔ **range(stop)**: exp: range(5) -> 0, 1, 2, 3, 4
- ➔ **range(start, stop)**: exp: range(5, 10) -> 5, 6, 7, 8, 9
- ➔ **range(start, stop, step)**: exp: range(5, 10, 2) -> 5, 7, 9

```
print(type(range(1,2)))          # ?

for i in range(8):
    print(i, end='')             # end??

for i in range(1, 6, 2):
    print(i, end=', ')

numbers = list(range(0,10))      # Cast range() to list
print(numbers)                   # ???
```

Exercises

Maktab Sharif

Practice

Maktab Sharif

Area & perimeter of Circle

Write a program that gets radius of the circle(x and y),
Then print area & perimeter of that.

exp:

```
>> Enter R: 12
>>
>> Area= 452.376
>> Perimeter= 75.396
```

Exercise: Wooooow!

Maktab Sharif

سوال زرد

- محدودیت زمان: ۰.۵ ثانیه
- محدودیت حافظه: ۲۵۶ مگابایت

مهدی که از کدزدن خسته شده‌است، به تازگی به رشته‌ی صنایع علاقه پیدا کرده است. به همین دلیل تصمیم گرفته است تا در مورد این رشته تحقیق کند. او به افراد مختلفی مراجعه می‌کند و هرکدام یک مقداری اطلاعات به او می‌دهند. او به اندازه‌ی مقدار اطلاعاتی که از اشخاص می‌گیرد متعجب می‌شود. مثلا اگر یک عدد اطلاعات بگیرد می‌گوید `Wow!` ، اگر دوتا اطلاعات بگیرد می‌گوید `Woow!` و به همین شکل مقدار کشیدن کلمه(تعداد `o` ها) زیاد می‌شود. حالا شما باید بگویید که اگر یک نفر به اندازه‌ی n به مهدی اطلاعات بدهد، ما باید انتظار چه کلمه‌ای را از او داشته باشیم.

Exercise: HW1 – Problem 5

Maktab Sharif

5- **(امتیازی)** برنامه ای بنویسید که در آن مقدار ثانیه را از کاربر بگیرید، سپس ساعت، دقیقه و ثانیه را در فرمت استاندارد در خروجی چاپ کنید. (راهنمایی: درمورد عملگر // و % تحقیق کنید)

نمونه ورودی:

```
>> 12345
```

نمونه خروجی:

```
>> 3:25:45
```

Example: say Hello to students

Maktab Sharif

```
# Write a program to saying hello to the all of students below

students_list = [
    'AmirHassan',
    'Sajjad',
    'Mehdi',
    'Hamid',
    "Rooh'o'lah",
    'Sina',
    'Mohammad',
    'Hamed',
    'Masoud'
]
```

while Example

Maktab Sharif

What's the output of program below:

```
n = int(input("Enter a number: "))

print(">> Start (n="+ str(n) +") <<")
while n:
    print(n * '*')
    n -= 1
print(">> End (n="+ str(n) +") <<")
```

Exercise

Maktab Sharif

> Simple calculator
> Write a simple calculator, that gets two Number and an operator from user.
> Then prints the result.

input:

```
>> 1st number: 11
>> 2nd number: -2
>> operator:   /
```

output:

```
>> -5.5
```

Example

Maktab Sharif

Big Vowels

Write a program that, gets a string

Then print a string with big Vowels (Uppercase vowels)

Input 1:

```
Hey akbar, How are you?
```

Output 1:

```
HEy AkbAr, HOw ArE yOU?
```

Input 2:

```
abcdefg
```

Output 2:

```
AbcdEfg
```

Example

Maktab Sharif

Reverse of number

Write a program that, gets a int number,
Then print the reverse of that

Input 1:

```
1423
```

Output 1:

```
3241
```

Input 2:

```
8830
```

Output 2:

```
388
```

Nested-loops example

Maktab Sharif

جدول ضرب گنده

برنامه‌ای بنویسید که با گرفتن n از ورودی جدول ضرب از ۱ تا n را چاپ کند.

input:

```
>> 6
```

output:

```
1       2       3       4       5       6
2       4       6       8       10      12
3       6       9       12      15      18
4       8       12      16      20      24
5       10      15      20      25      30
6       12      18      24      30      36
```

Exercise: Single digit

Maktab Sharif

تک‌رقمی

مهدی که از کدزدن خسته شده‌است، دیگر حوصله اعدادی که بیشتر از یک رقم دارند را ندارد. به همین خاطر به هر عدد چند رقمی که بر بخورد آن را به شیوه خاص خودش تبدیل به یک عدد تک رقمی می‌کند. به این شکل که عدد مورد نظر را با عدد حاصل از مجموع ارقام آن جایگزین می‌کند و به یک عدد جدید می‌رسد. سپس همین کار را با عدد جدید انجام می‌دهد و تا جایی که به یک عدد تک‌رقمی برسد به این کار ادامه می‌دهد. بعد از مدتی مهدی متوجه شد که با این کار نه تنها راحت تر نشده است، بلکه بیشتر درگیر اعداد شده است. در نتیجه از شما خواسته است در یک رقمی کردن عددها به او کمک کنید.

ورودی نمونه ۱

14

خروجی نمونه ۱

5

ورودی نمونه ۲

123456

خروجی نمونه ۲

3

Pre-reading

Search about:

1. String methods
2. Tuple in python
3. Set in python
4. Tuple vs. list
5. Set vs. list
6. Function in python
7. sum(), min(), max() functions
8. Lambda functions in python
9. map() in python