



Python Programming









Session Objective



To understand the basic concepts of Python,

Control Structures

-  If statement
-  If-else statement
-  If-elif-else statement
-  For Loop
-  While Loop
-  Break, Continue & Pass

Input and Output

String Function

Number Function

Date and Time Function



Control Structures



Control Structures

There are three important control structures

Sequential

Alternative or
Branching

Iterative or Looping

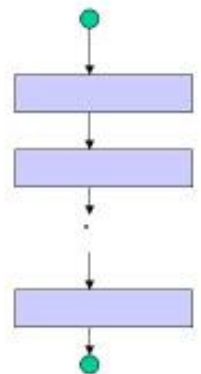
Control Structures

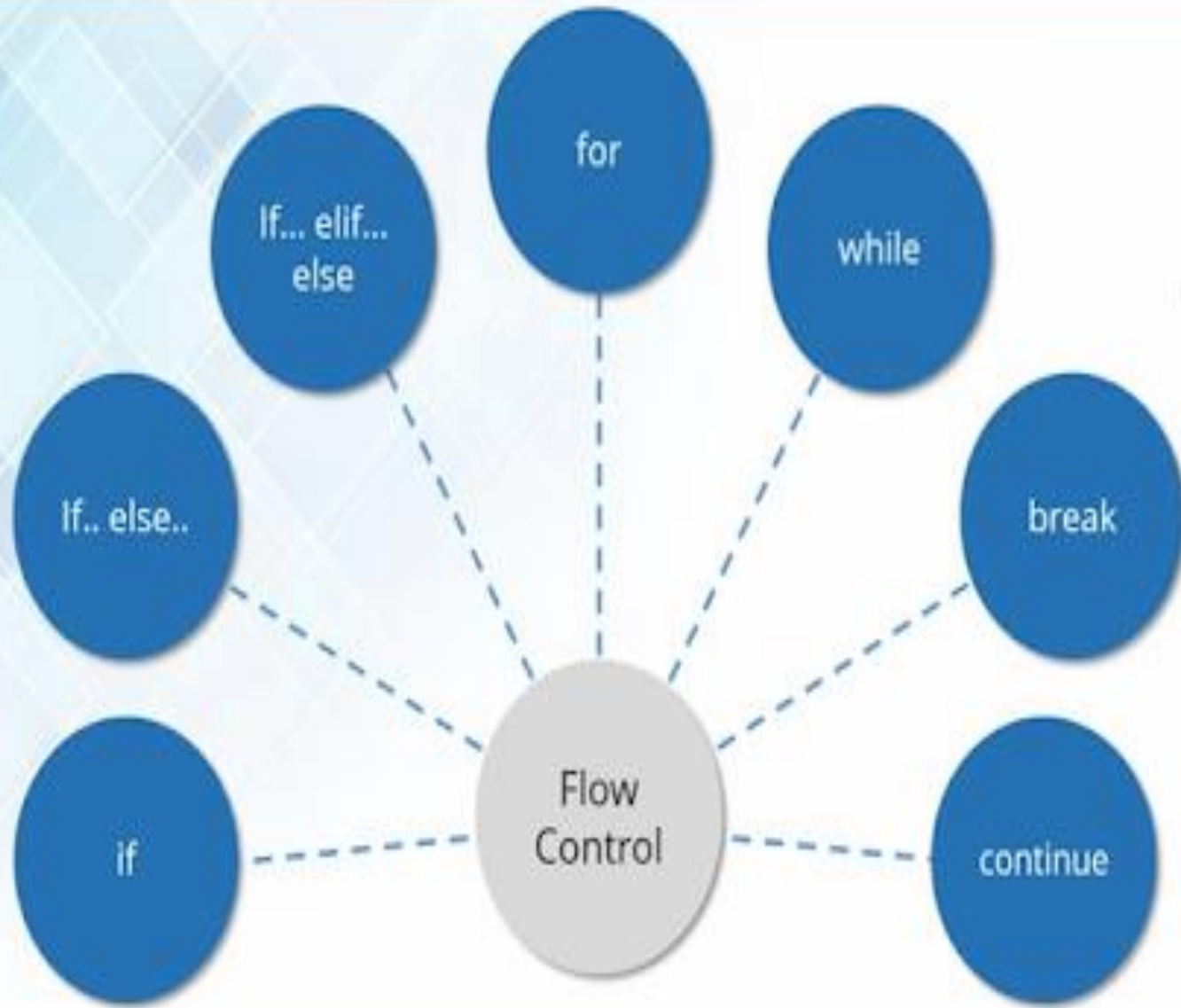
- Control structures control the flow of program execution.
- 3 types of control structures: sequence, selection repetition

- A sequence control structure uses compound statements (blocks) to specify sequential flow.
- A compound statement, a block of codes, is used to specify sequential flow.

```
{  
    statement1;  
    statement2;  
    statement3;  
}
```

Sequence





Control Structures

Contd...



IF and IF-ELSE Statement

- ❑ Decision making is used to execute a code only if a certain condition is satisfied.
- ❑ Python uses indentation for grouping statements.

IF	IF-ELSE
<pre>num = 45 # only if if num > 25: print("Hurray! {} is greater than 25".format(num))</pre>	<pre>x=8 r=x % 2 if r==0: print("Even") print("good") else: print("Odd")</pre>



Nesting of IF

- ❖ Block of code for functions, control structures, etc are distinguished by indented code
- ❖ 4-space indentation is recommended
- ❖ A common syntax error is leaving out : at end of control structure statements
- ❖ Using () around conditions is optional
- ❖ Indented block can have any number of statements, including blank lines

```
x=8
r=x % 2
if r==0:
    print("Even")
    print("good")
    if True:
        print("Great")
else:
    print("Odd")
```



IF-ELSE-IF Statement

- ❑ The keyword 'elif' is short for 'else if' and is useful to avoid excessive indentation.
- ❑ An if ... elif ... elif ... sequence is a substitute for the switch or case statements found in other languages.

```
a=2
b=15
c=8
if a>b:
    print("a:",a)
elif a>c:
    print("a with c:",a)
elif b>c:
    print("b with c:",b)
else:
    print(c)
```




For Loop

For Loop Syntax	For Loop Example	Output
for iterator_var in sequence: statements(s)	words = ['mac', 'window', 'linux'] for w in words: print(w, len(w))	mac 3 window 6 linux 5
	number = 9 for i in range(1, 5): mul_table = number * i print("{} * {} = {}".format(number, i, mul_table))	9 * 1 = 9 9 * 2 = 18 9 * 3 = 27 9 * 4 = 36



While Loop

While Loop Syntax	While Loop Example	Output
while expression: statement(s)	count = 0 while count < 5: print(count) count += 1	0 1 2 3 4



Break Statement

Break brings control out of the loop

Break	Output
<pre>for letter in 'Python': if letter == 't': break print('Current Letter :', letter)</pre>	Current Letter : t



Pass Statement

- ❑ Pass statement are used to write empty loops.
- ❑ Pass is also used for empty control statement, function and classes.

Pass	Output
<pre>for letter in 'Python': pass print('Last Letter :', letter)</pre>	Last Letter : n



Continue Statement

It returns the control to the beginning of the loop.

Continue	Output
<pre>for letter in 'Python': if letter == 'o': continue print('Current Letter :', letter)</pre>	<pre>Current Letter : P Current Letter : y Current Letter : t Current Letter : h Current Letter : n</pre>



Input and Output

Input and Output



Input from console

Console is also called Shell; this is basically a command line interpreter that takes input from the user i.e one command at a time and interprets it

STEP 1:

Create a file First.py

STEP 2:

Enter the following code

```
x=input("Enter 1st num:")  
a=int(x)  
y=input("Enter 2nd num:")  
b=int(y)  
z=a+b  
print(z)
```



STEP 3.A:

- ✓ Open the Command prompt
- ✓ Execute the command in the Python's file path

```
C:\Users\31410\Desktop\Python_Demo>python First.py
Enter 1st num:33
Enter 2nd num:44
77
```



Input from PyCharm IDE

- ✓ To execute in IDE, open PyCharm
- ✓ Run the file

A screenshot of the PyCharm IDE interface. The top menu bar includes File, Edit, View, Navigate, Code, Refactor, Run, Tools, VCS, Window, and Help. The title bar shows 'SampleProject [C:\Users\31410\PycharmProjects\SampleProject] - ...\Welcome.py - PyCharm'. The left sidebar shows the Project view with 'SampleProject' expanded, containing 'venv library root' and 'Welcome.py'. The main editor shows 'Welcome.py' with the following code:

```
1 x=input("Enter 1st num:")
2 a=int(x)
3 y=input("Enter 2nd num:")
4 b=int(y)
5 z=a+b
6 print(z)
```

The bottom panel shows the Run view with the command 'C:\Users\31410\PycharmProjects\SampleProject\venv\Scripts\python.exe C:/Users/31410/PycharmProjects/SampleProject/Welcome.py'. The output shows 'Enter 1st num:22', 'Enter 2nd num:33', and '55'. The status bar at the bottom indicates 'Process finished with exit code 0'.



String Functions





String

- ❖ Strings are sequences of characters and are immutable.
- ❖ Python string is the collection of the characters surrounded by single quotes, double quotes, or triple quotes.

Creating String in Python

- ❑ String can be created by enclosing the characters in single-quotes or double- quotes.

```
#Using single quotes  
str1 = 'Hello Python'  
print(str1)
```

```
#Using double quotes  
str2 = "Hello Python"  
print(str2)
```



- ❑ Python also provides triple-quotes to represent the string, but it is generally used for multiline string or docstrings.

```
#Using triple quotes  
str3 = """Triple quotes are generally used for  
represent the multiline or  
docstring"""  
print(str3)
```



- ❑ String can also be created with the str() function.

```
>>> str(123)
'123'
```

```
>>> str('programming')
'programming'
```

```
>>> str(programming)
```

Traceback (most recent call last):
File "<pyshell#1>", line 1, in <module>
str(programming)
NameError: name 'programming' is not
defined



String Indexing:

The indexing of the Python strings starts from 0.

P	Y	T	H	O	N
0	1	2	3	4	5

String

Index

```
>>>str="PYTHON"
>>>print(str)
PYTHON

>>>print(str[1])
Y
>>>print(str[7])
IndexError: string index out of range
```



String Slicing

Parts of strings can be retrieved at one time by slicing it with the slicing operator [:]

```
Str="programming language"  
>>> str="PYTHON"  
>>> print(str[2:5])  
Output  
THO
```

[2:5] is a slice that gives us the characters from index 2 to index 4, index 5 is not included.

```
>>> print(str[:])  
Output  
PYTHON
```

Prints the entire String



String Slicing

```
>>> print(str[2:])  
THON
```

Executes from 2nd index to the end of the string

```
>>> print(str[-2:])  
ON
```

Negative index will start from the end of the string

```
>>> print(str[-4:-2])  
TH
```

Executes 2 character from 4th index tracing from the end of the string



Concatenation:

Concatenation is joining two strings with the “+” operator.

```
>>> 'python'+ 'program'
'python program'
```

```
>>> 'python '*3
'python python python '
```

```
>>> 'python'+3
```

cannot concatenate a string with an integer.

Traceback (most recent call last):
File "<stdin>", line 1, in <module>
TypeError: can only concatenate str
(not "int") to str



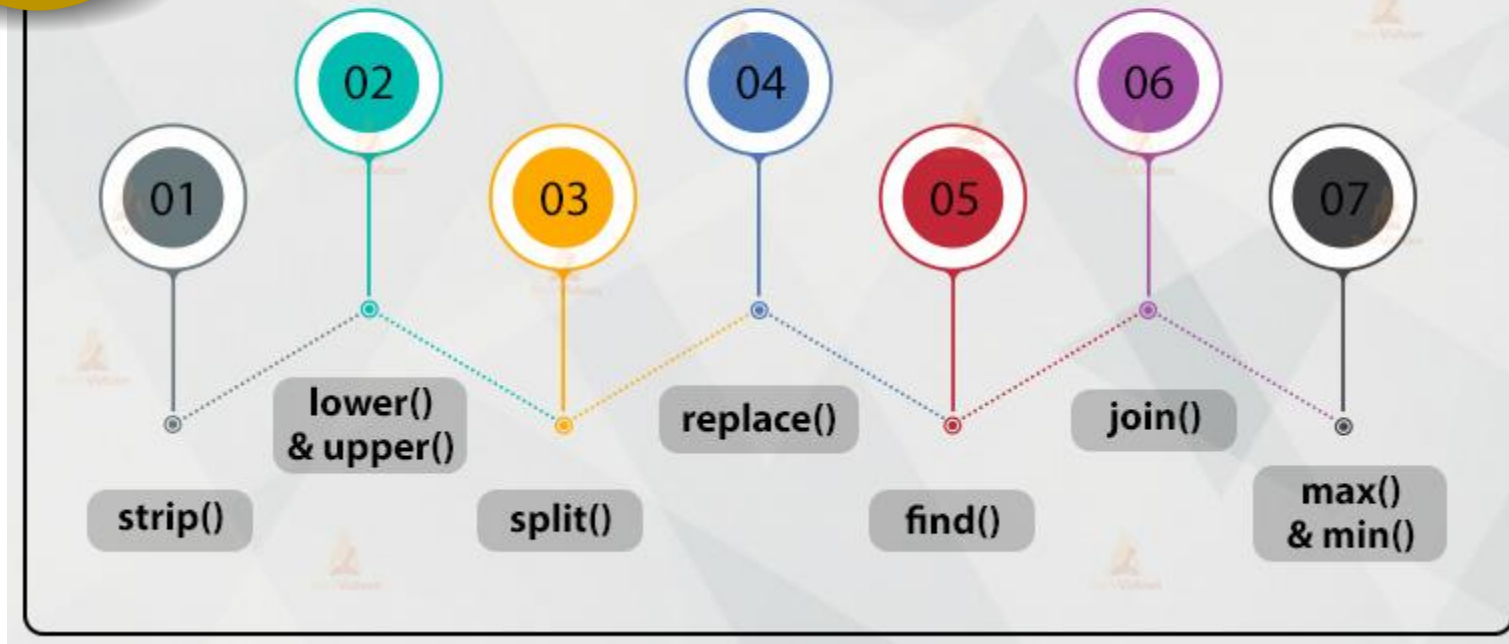
String Length

To find the length of the string, len() function is used

```
>>> str="Python is a Programming Language"  
>>> len(str)  
32
```



PYTHON STRING METHODS & FUNCTIONS





Strip()

The strip() method removes whitespace around the string and returns the rest.

```
>>> str=" String Functions "  
>>> print(str)  
String Functions
```

String with whitespace

```
>>> print(str.strip())  
String Functions
```

String without whitespace



Upper(), Lower() & Replace()

```
>>> str="String function"
>>> print(str.upper())
STRING FUNCTION

>>> print(str.lower())
string function

>>> print(str.replace('function','methods'))
String methods
```

Executes the given string in upper case

Executes the given string in lower case

Replace the 1st argument by 2nd argument



Split() & Find()

- split() method returns a list of strings after breaking the given string by the specified separator.
- The find() method returns the first index where it finds a character.

```
>>>str="String function"

>>> print(str.split(' '))
['String', 'function']

>>> print(str.find('m'))
-1
>>> print(str.find('t'))
1
```

➤ If it does not find the character, it returns -1.



Max() and Min()

The max() and min() methods gives the characters with the highest and lowest values.

```
>>> str="pyz"  
  
>>> print(max(str))  
z  
  
>>> print(min(str))  
p
```



Number Functions



Number Type Conversion

Python converts numbers internally in an expression containing mixed types to a common type for evaluation

- Type `int(x)` to convert `x` to a plain integer.
- Type `float(x)` to convert `x` to a floating-point number.
- Type `complex(x)` to convert `x` to a complex number with real part `x` and imaginary part zero.



Mathematical Functions

- The math module is a standard module in Python.
- To use mathematical functions import the module using import math.

```
>>> import math

>>> x=math.sqrt(25)
>>> x
5.0

>>> x=math.sqrt(15)
>>> x
3.872983346207417
```



Mathematical Functions

```
>>> print(math.floor(2.9))
2
>>> print(math.floor(2.2)) [least value]
2
>>> print(math.ceil(2.2)) [highest value]
3
>>> 3**2
9
>>> print(math.pow(3,2))
9.0
>>> print(math.pi)
3.141592653589793
```



Date and Time Function



Time

- ✓ Python's time and calendar modules help to track dates and times.
- ✓ The function `time.time()` returns the current system time in ticks since 12:00am, January 1, 1970(epoch).
- ✓ Import the time module

```
>>> import time;

>>> print(time.time())
1594894062.8836365

>>> print(time.localtime())
time.struct_time(tm_year=2020, tm_mon=7, tm_mday=16, tm_hour=15,
tm_min=41, tm_sec=16, tm_wday=3, tm_yday=198, tm_isdst=0)
```



Current Time

To translate a time instant from seconds since the epoch floating-point value into a timetuple, pass the floating-point value to a function (e.g., `localtime`) that returns a time-tuple with all valid nine items.

```
import time

>>> currentTime=time.localtime(time.time())

>>> print("Local current time is:",currentTime)

Local current time is: time.struct_time(tm_year=2020, tm_mon=7,
tm_mday=16, tm_hour=16, tm_min=28, tm_sec=36, tm_wday=3,
tm_yday=198, tm_isdst=0)
```



Formatted Time

Time can be formatted as per our requirement, but a simple method to get time in a readable format is `asctime()`

```
>>> currentTime=time.asctime(time.localtime(time.time()))
```

```
>>> print("Local time:",currentTime)
```

```
Local time: Thu Jul 16 16:33:46 2020
```



Calendar

- ❑ To print a calendar for a given month
- ❑ Import the Calendar module

```
>>> import calendar
```

```
>>> cal=calendar.month(2020,7)
```

```
>>> print(cal)
```

```
July 2020
Mo Tu We Th Fr Sa Su
 1  2  3  4  5
 6  7  8  9 10 11 12
13 14 15 16 17 18 19
20 21 22 23 24 25 26
27 28 29 30 31
```



```
#Import the datetime module and display the current date:
```

```
x = datetime.datetime.now()
```

```
print(x)
```

```
#The strftime() method returns a string representing date and time using date, time or datetime object.
```

```
print(x.year)
```

```
print(x.strftime("%A"))
```

```
year = x.strftime("%Y")
```

```
print("year:", year)
```

```
month = x.strftime("%m")
```

```
print("month:", month)
```



```
day = x.strftime("%d")
print("day:", day)

time = x.strftime("%H:%M:%S")
print("time:", time)

date_time = x.strftime("%m/%d/%Y, %H:%M:%S")
print("date and time:", date_time)

#Create a date object:
x1 = datetime.datetime(2020, 5, 17)
print(x1)
```

Think and Answer

1. What is the output of the following code?
`print(int(2.999))`
2. What is the output of "hello"+1+2+3?
3. Which function return date and time in particular format?
4. Which function is used to pause the code for the specified number of seconds?
5. String is Mutable or immutable?



Think and Answer



RIGHT ANSWER

1. 2
2. Error
3. `asctime()`
4. `Sleep()`
5. `immutable`



Thank you

Innovative Services



Passionate Employees

Delighted Customers

