Unit 12
Built-in Functions in Python

Table of Contents

12.1 Introduction
Learning Objectives
12.2 Date and Time
12.2.1 time.time()
12.2.2 time.localtime()
12.2.3 asctime()
12.2.4 sleep()
12.2.5 Creating date objects
12.2.6 Comparison of two dates
12.2.7 Calendar module
Self-Assessment Questions
12.3 Miscellaneous
12.3.1 abs()
12.3.2 all()
12.3.3 bin()
12.3.4 bool()
12.3.5 bytes()
12.3.6 callable()
12.3.7 compile()
12.3.8 exec()
12.3.9 sum()
12.3.10 any()
12.3.11 ascii()
12.3.12 help()
Self-Assessment Questions
12.4 Summary
12.5 Glossary
12.6 Case Study
12.7 Terminal Questions
Answer keys
A. Self-Assessment Questions
B. Terminal Questions
12.8 Suggested Books and e-References

12.1 INTRODUCTION

Python programming is used widely around the globe on different softwares, development and management of data. You learnt some of the basic blocks of the Python programming language that can be used to code a program. In the previous section, you were introduced to exception handling in Python. You may learn to write a code or a program to any problem presented to you, but it is equally important to learn to handle the errors and exceptions generated by Python itself.

These errors or exception can be related to anything in your code. The errors generated by Python during the execution of the program refer to exception. An exception is due to some internal issue in the program which obstructs its execution. Exception handling can be done by using statements such as try, except, and so on. Using these statements you can easily remove the barriers in your code and make the execution smooth and uninterrupted.

What are built-in functions in Python?

Python programming language comes with an impressive and huge standard library that allows the programmers to use the built-in functions present in their program to carry out the specific tasks. This makes programming in Python simple and straightforward. It saves a lot of time, energy and data for the programmers.

The built-in functions in Python are those functions whose tasks are already pre-defined in Python. The programmers have to need to define such functions. These functions can be used directly in any program as deemed necessary by the programmers. Python has a total of 67 built-in functions so far for the convenience of the programmers.

Advantages of built-in functions in Python

There are several advantages of built-in functions in the Python programming language, some of which are given below.

- It reduces the length of the code.
- It makes a code easy to understand.
- The code becomes reusable.
- Complex programs can be solved easily in parts.

Since there are so many built-in functions in the standard library of the Python programming language, it is not possible to explain every one of them. You already know some basic built-in functions like print(), max(), min(), etc. But we have assembled some important built-in functions in this unit that might come in handy frequently.

LEARNING OBJECTIVES

After studying this chapter, you will be able to:

- Understand the concept of built-in functions in Python.
- Explain the date and time functions.
- Describe the method of creating date objects.
- Elaborate on how do you compare two dates.
- Explain the calendar module.
- Understand the role of miscellaneous built-in functions.

12.2 DATE AND TIME

Time is a complex and crucial factor. It is one of the foundations of a structured world and universe. Without something like time to keep in check the events, the world will erupt into chaos and disorder. Not a single platform, organisation or country could move forward without order and structure. Therefore it is necessary to know how to implement date and time in any software, or service that you would be working at.

Python programming language has many built-in libraries in which date and time module are also present. Several built-in functions are present in the time module that are used to carry out different operations using time. These functions can do the conversion, representation and several other implementations that are necessary for life. Here are some terms listed out that you need to understand before knowing the functions.

To carry out these operations, we must be aware of the starting point. Epoch in Python corresponds to the measurement of time starting from January 1, 1970. Epoch is generally known as the starting point of the time.

The total number of seconds passed since epoch is referred to as seconds since the epoch. Leap seconds are not included in seconds since the epoch.

UTC is referred to as Coordinated Universal Time. This same term was known as Greenwich Mean Time or GMT.

The times presented in the output by these functions are in floating numbers. Let us discuss the functions from the time module one by one.

12.2.1 time.time()

The time.time() function returns the number of seconds passed after epoch. Suppose one needs to finds out the number of seconds passed since epoch(the beginning of time), then this function comes in handy. Simply apply the time.time() and get the results in a floating number. The syntax for this particular function is given below.

```
time.time()
```

An example of this can be seen below.

```
#applying the time.time() function and print the return
import time
seconds passed since epoch = time.time()
print(seconds passed since epoch)
```

In this particular example, the time module is imported from the built-in libraries of the Python programming language. The return of the function is stored and then printed to know the value of output. This is the most basic function of the time module.

12.2.2 time.localtime()

The function, time.localtime(), is used to convert the number of seconds elapsed since epoch and returns the value in local time using the function struct_time(). The syntax for the function is given below.

```
time,localtime([secs])
```

The parameter required for this function is optional. The parameter is in the form of integer or floating value. If secs is not specified for the function, then the current time will be calculated using the time.time() function. The return of this function is an object of class time.struct_time. An example for the above function is given below.

```
#applying the time.local time() function and print the return
import time

#no parameter will be provided in this example
object = time.localtime()

#print the object to verify the return type
print(object)
```

In this particular example, after importing the time module to use its function, an object is created to get the return in the localtime and the return is obtained by using the print function.

Since we get the output in the form of values returned through time.struct_time, it is important we know all the values of attributes used and know the terms more clearly. The table given below presents all the values in order.

Index	Attributes	Values
0	tm_year	example -1995, 2003 or 2015
1	tm_mon	1, 2, 3, ….., 12
2	tm_mday	1, 2, 3, ….., 31
3	tm_hour	0, 1, 2, ….., 23
4	tm_min	0, 1, 2, ….., 59
5	tm_sec	0, 1, 2, ….., 61
6	tm_wday	0, 1, 2, ….., 6; Monday: 0

7	tm_yday	1, 2, 3, ….., 366
8	tm_isdst	0, 1, or -1

One can use both the attributes and the indices to access the above values for programs.

12.2.3 asctime()

The time.struct_time and the time.localtime() return values in tuple form. The asctime() function is used to convert the tuple returned from these functions to string form. The syntax can be seen below.

```
time.asctime([t])
```

the parameter t represents a tuple or a time.struct_time object that is returned by either time.localtime() or time.gmtime(). This is entirely optional and the default value of this parameter is the current time given by time.localtime(). The return type of the following function is presented in the form below.

```
Day Mon Date Hour:Min:Sec Year
```

An example is presented below for a better understanding of the function.

```
#time module is imported first
import time

#parameter t is not given
#current time is given by time.localtime()

#conversion of current time (tuple) returned by time.localtime() to the form(string) Day Mon Date Hour:Min:Sec Year
str = time.asctime()

print(str)
```

The output of the given code is presented below.

```
Thu Apr 01 12:16:23 2021
```

This is a very good example to understand the function and working of this time function. The values that we obtain through time.struct_time object can be easily converted by the use of this simple function.

12.2.4 sleep()

During the execution of the program on any device, sometimes the flow of the program needs to stop for time being. The time is specified and for this purpose, the sleep() function is used.

It is a flexible and convenient way of halting a program for specific needs for the desired period. the syntax for this function is given below.

```
sleep(sec)
```

The parameter sec refers to the number of seconds the program has to be suspended. This parameter is necessary because, without it, Python will not know how many seconds the program has to be halted. The return for this function is nothing, void. It may seem like a useless function but it has any uses that will get know by looking at the examples given below.

```
#importing the time module
import time

#printing a sentence
print(“This sentence is printed right now.”)

#printing a sentence after 10 seconds
time.sleep(10)
print(“This sentence is printed after 10 seconds.”)
```

In this program, the first sentence will be printed immediately. however, there will be a delay of 10 secs before printing the second sentence. Let's look at some applications of this built-in function.

12.2.5 CREATING DATE OBJECTS

What is a date? A date is simply a representation of a day, month and year. Creating a date object means an object which represents a day, month and year. We can take the help of the date class to create a date object. Take a look at the examples given below to understand better.

```
#importing the time module
import time

#creating a date object x
x = datetime.date(2021, 1, 4)

#printing the variable x
```

```
print(x)
```

The output will be as follows.

```
2021-01-04
```

In this example, the date object is created from the date class and stored in a variable. The date contains three arguments, that is, day, month, and year. you have to note that the variable x here is the date object that is created using the date class and the date constructor.

We can also import the date class from the DateTime module as demonstrated below.

```
#importing the date class from the DateTime module
from DateTime import date

#creating a date object y
y = date(2021, 2, 4)

#printing the output
print(y)
```

There is also a second method you can use for creating the date objects to get the current date. You can take the help of the classmethod named today() for that purpose. Let us take a look at how can we do it.

```
#importing the date class from the DateTime module
from datetime import date

t = date.today()

print(“The current date”, t)
```

In this example, the output will be the current date.
Do you know what is timestamp is? A timestamp refers to the number of seconds between January 1, 1970, and the desired date. Why I am referring to this term is we can also create the date object from a timestamp. You will understand the concept more by looking through the following example.

```
#importing the date class from the datetime module
from datetime import date

tstamp = date.timestamp(1374839227)
print(“The date required is =”, tstamp)
```

The output will the date till which are the seconds measured from January 1, 1970.

You can create date objects with these methods as required in your program. You must not forget to import the modules require or Python will generate an error because it will not recognize the functions.

12.2.6 COMPARISON OF TWO DATES

How do you compare two things? Obviously, by the help of comparison operators. We have learnt about the comparison operators in unit 10. The comparison operators are <, >. =, <=, >=, !=, and so on. Can you recall what the most basic condition to compare two objects is? They should belong to the same group. For example, you cannot compare a string and a number, or a list or a number. Comparison is done between the elements of the same group.

STUDY NOTE

The date objects can be compared the same way the dates are compared.

Let us look through some examples to know more about the comparison of the two dates.

```
#import datetime module
import datetime

#create dates in the format (dd/mm/yyyy)
x = datetime.datetime(2021, 3, 2)
y = datetime.datetime(2021, 5, 2)

#comparison of the dates
a = x > y
b = x < y
c = x != y

#print the values
print("x is greater than y:", a)
print("x is less than y:", b)
print("x is not equals to y:", c)
```

The output of the above program will be the boolean values and is demonstrated below.

```
x is greater than y: TRUE
x is less than y: FALSE
x is not equal to y: TRUE
```

You can also sort the dates and put them into a list or tuple. Comparison operators make this easy.

12.2.7 CALENDAR MODULE

This in-built calendar module includes all the functions and classes that take care of the operations related to a calendar. The calendar is different in different regions, but the one recognized by the Python programming language is the current Gregorian calendar in which Monday is considered as the first day of the week and Sunday, the last day of the week. The functions and classes included in the calendar module are essential for the implementation of the functions on the calendars. There are three classes in total included in the calendar module.

- Calendar
- TextCalendar
- HTMLCalendar

The Calendar Class

After going through the following examples, we will get a better understanding of how to use the classes and functions in the calendar module. in this first example. We will write a program to display a calendar of a particular month.

```
#importing the calendar module
import calendar

#mentioning the year and month of which we want the calendar

yy = 2021
mm = 04

#printing the calendar
print(calendar.month(yy, mm))
```

The output will be displayed, that is April of the year 2021.

You can print the calendar for the whole year in the same way.

Creating calendar objects

You can create a calendar object from the calendar class. The process is the same only the class and object is different. Several formatting methods can be used through the calendar object. Some of the useful methods from the calendar class are given below.

- **iterweekdays() Method**

A list of indexes is returned for the days of the week by this method. a simple example given below will allow you to understand more easily.

```
#importing the calendar module
import calendar

cal = calendar.Calendar()
for i in c.iterweekdays():
    print (i, end="")
```

The output is as follows.

```
0 1 2 3 4 5 6
```

- **itermonthdates() Method**

Two arguments are required for this method, namely, year and month. Iterator of all the days of the month is returned by the use of this method. Look through the given example.

```
#importing the calendar module
import calendar

cal = calendar.Calendar()
for i in c.itermonthdates (2021, 3):
    print (i, end="")
```

- **itermonthdays() Method**

This method returns the day numbers and is similar to the itermonthdates method.

```
#importing the calendar module
import calendar

cal = calendar.Calendar()
for i in c.itermonthdays (2021, 3):
    print (i, end="")
```

The output is as follows.

```
0, 1, 2, 3, 4, 5, 6, 7, ……., 30, 31, 0, 0, 0
```

- **monthdatescalendar() Method**

Two arguments, year and month, are required for this method. This method returns full weeks in a month.

```
#importing the calendar module
import calendar

cal = calendar.Calendar()
for i in c.monthdatescalendar (2021, 3):
    print (i, end="")
```

- **monthdayscalendar() Method**

Two arguments, year and month, are required for this method. This returns a list of full weeks and each week containing the list of days of a month.

- **yeardatescalendar() Method**

This method returns the list of weeks in the month of the year as full weeks.

The TextCalendar Class

This class is used to generate text calendars. It is very similar to the class calendar. Some of the methods in TextCalendar Class are given below for a better understanding of the concept.

- **formatmonth() Method**

Four arguments are required for this methods: year, month, the width of days(w), and the number of lines used for each week(1). Multi-string is returned by this method. An example of this method is presented below.

```
#importing the calendar module
import calendar

cal = calendar.TextCalendar()
print(c.formatmonth(2021, 1))
```

The output will be the calendar for the given month and the year.

- **prmonth() Method**

This method is used to avoid the use of the print() function and returns the month of the calendar.

- **formatyear() Method**

An "m" column calendar for the entire year is returned by this method in the form of a multi-line string.

The HTMLCalendar Class

This class generates an HTML calendar. Some methods that are included in the HTMLCalendar Class are presented below.

- **formatmonth() method**

The calendar of the month in HTML table format returned from this method. we can easily display a calendar for a month for any year using this method.

- **formatyear() method**

This method prints the calendar for the entire year, whichever desired by the programmer or the user, in the HTML table format. Year and number of months are taken in as the argument for this method. The default width is set to 3.

- **formatyearpage() method**

This also printsthe calendar for the entire year, whichever desired by the programmer or the user, in the HTML table format. The difference is that this method also takes CSS and encoding as arguments which are optional as well. The default value of CSS and encoding is set to none and width is set to 3.

ACTIVITY 1

Understand the above functions and use the sleep() function of the time module to make a digital clock. Make sure to print the local time after each second. This means after every print, a delay of 1 second has to be made.

SELF-ASSESSMENT QUESTIONS

1) The built-in functions in Python are those functions whose tasks are already -____________ in Python.

2) __________is used to convert the number of seconds elapsed since epoch.

3) The date objects can be compared the same way the dates are compared. (True/False)

4) The Calendar module consists of three classes. (True/False)

5) The HTML Calendar class contains

   A. formatmonth() method
   B. formatyear () method
   C. formatyearpage() method
   D. All of these

12.3 MISCELLANEOUS

There are numerous built-in functions in various classes and methods, each one useful in its way. Still some functions are used for several different tasks and they are discussed below one by one.

12.3.1 abs()

This function is used to present the absolute value of the given number. The syntax for the abs() function is given below.

abs(num)

The parameter num (which can be a floating-point number, complex number, or an integer) is given by the programmer to the function. abs() will then take out the absolute value of the number and display the output to the programmer.

For integers and floating-point numbers, the function displays the absolute value. In the case of complex numbers, it returns the magnitude part of the complex number to the user.

An example of the use of the function is presented below. We will present all the numbers in the code for a better understanding of the function.

```
#enter a floating point number
float =  -87.03

#print the absolute value
print("The absolute value of the floating number is:", abs(float))

#enter an integer
integer =  67

#print the absolute value
print("The absolute value of the integer is:", abs(integer))

#enter a complex number
complex =  4 + 3j

#print the absolute value
print("The absolute value of the complex number is:", abs(complex))
```

```
The absolute value of the floating number is: 87.03
The absolute value of the integer is: 67
The absolute value of the floating number is: 5.0
```

12.3.2 all()

The all() function returns the boolean value

- TRUE if all the present elements are TRUE and
- FALSE if all the present elements are FALSE

The syntax of all() function is given below.

```
all(iterable)
```

As shown, the parameter used can be the elements of a list, tuple, dictionary, or string.

Truth Table for all() function:

When	Return Value
All are true	TRUE
All are false	FALSE
Only one is true	FALSE

Only one is false	FALSE
Empty	TRUE

Example of all() function for strings:

```
#create a set
set1 = {1, 0, 0, 0 ,1}
a = all(set1)
print(a)
```

The output of the above code will be boolean value FALSE.

12.3.3 bin()

Conversion is an important operation and is applicable every day. Now and then we have to convert something for our convenience or the system's convenience. The bin() function converts the decimal number into a binary number which is a very important task in Python. This function comes in handy when we want to do a quick conversion for any program or code. it reduces the length of the code and any complex codes that are required for this conversion.

STUDY NOTE

Note that every return value of bin() is joint with prefix 0b.

The syntax for this function is given below.

```
bin(num)
```

The num parameter can be any integer that has to be converted to a binary number for some operation. The return value is in the type of a binary string of integer values. Error is raised when a floating-point number is given in the argument. Let us look at an example for this function.

```
#function returning string
def binary(a):
    x = bin(a)
print(“The binary number obtained from the conversion is:”, binary(2))
```

The output of the following code is given below.

```
0b10
```

12.3.4 bool()

The bool() functions either return a boolean value or converts the given value to a boolean value. The syntax for the boolean value is given below.

```
bool([x])
```

This function requires a single parameter that is evaluated to give the results awaited. If nothing is passed as a parameter to the function then the default value, that is, FALSE is printed. The function does one of the two things after receiving the parameter:

- It prints TRUE if the parameter given by the programmer is TRUE.
- It prints FALSE if the parameter given by the programmer is FALSE.

Some of the conditions for which this function returns FALSE is listed below. Other than these, it returns the boolean value TRUE.

- nothing is passed as a parameter.
- a false value is passed to the function.
- zero is passed in any form to the function.
- empty values are passed such as (), {}, [], etc.
- objects of classes returning 0 or FALSE.

Take a look at the example given below.

```
#create string
str = “I can code very well.”
print(bool(str))
```

The output of the above code is given below.

```
TRUE
```

12.3.5 bytes()

Interconversion of data types in Python can be done by using the bytes() function. It is useful for converting data types using its coding schemes. it converts the given object into an immutable object of the right size and data. The syntax for the bytes() function is given below.

```
bytes(source, encoding, errors)
```

The parameter source is the object which is to be converted. Encoding is the parameter that is required for the object in the string. The error provides the method to handle the errors generated if the conversion is not successful. It returns an immutable object in Unicode characters according to the source type. Look at the example given below.

```
#create a string
str = “Python is the best”
x = bytes(string, ‘utf-8’)
 print (x)
```

The output for the above code is given below.

```
CPython is the best.'
```

12.3.6 callable()

Callable is a term that means something that can be called. In Python, the callable() function returns the boolean value TRUE if the argument appears callable and FALSE if it does not. The syntax for this function is given below.

```
callable(object)
```

The object passed into the function is checked whether it is callable or not and then the corresponding boolean value is returned.

Check out the below example to know the task of this function.

```
def func():
   return 10

# an object is created of func()
let = func
print(callable(let))

# a test variable
num = 5 * 10
print(callable(num))
```

The output is given below.

```
TRUE
FALSE
```

12.3.7 compile()

Can you remember when you used to compile the source code in C language and then the code was executed? This function does the same thing. It takes in a source code and returns a code object which will be ready for execution. The syntax for this function is presented below.

```
compile(source, filename, mode, flags=0, dont_inherit=False, optimize=-1)
```

The parameters are discussed below.

- the source can be a string, a byte string, or an AST object.
- the filename is the file from which you read the code and name you can give yourself.
- The mode can be either eval, exec, or single.
  Eval: single expression containing the source.
  Exec: a set of code containing statements, functions, class, and so on.
  Single: a single Python statement
- Flags are optional and take care of the future statements that are likely to affect the compilation of the source.
- Optimize(optional) provides the optimization level of the compiler.

Here is an example of this function.

```
x = 20

# eval is used for single statement
a = compile('x', 'test', 'single')
exec(a)
```

The output is as follows.

```
20
```

12.3.8 exec()

This function in Python is responsible for the dynamic execution of the program for either string or object code. the syntax for these functions is given below.

```
exec(object, globals, locals)
```

The object parameter can be a string or object code. Globals is optional and can be a dictionary. Locals are also optional and can be a mapping object.

```
prog = 'print("The sum of 3 and 17 is", (3+17))'
exec(prog)
```

The output is as follows.

```
20
```

12.3.9 sum()

This function is the most used and common. It returns the sum of the elements(numbers) in a list, tuple, dictionary, or any set. the syntax is presented below.

```
sum(iterable, start)
```

The iterable refers to the elements present in the set, list, tuple, or dictionary. The tables should be numbers only. The start is optional and is added to the sum of the iterables. The default value is 0 if anything is not given. The example is given below.

```
numbers = [1,2,5]

Sum = sum(numbers)
print(Sum)

Sum1 = sum(numbers, 10)
print(Sum1)
```

The output is given below.

```
8
18
```

12.3.10 any()

The any() returns the boolean value TRUE for even a single true element in the iterable and FALSE if all the elements in the iterable are false. The syntax for the given function is presented below.

```
any(iterable)
```

The iterable can be a set, list, dictionary, or tuple. Let us look at the given example.

12.3.11 ascii()

This function returns a string of representation of the object and escapes the non-ASCII characters in the string using \x, \u or \U escapes. The syntax of the function is given below.

```
ascii(object)
```

The object is iterable of which printable representation is returned.

For example for the input: ascii("µ")
Output : '\xb5'

12.3.12 help()

This function is used to display the documentation of the functions, keywords, module, or classes, etc. The syntax of the function is given below

help([object]

SELF-ASSESSMENT QUESTIONS

6) The ____________function is used to convert the tuple returned to string form.

7) The____________ function converts the decimal number into binary number.

8) The help() function gives a list of instructions to do when any error is generated. (True/False)

9) The sleep() function halts the execution of the sometime. (True/False)

10) Which of the following is not a built-in function
   A. all()
   B. help()
   C. minimum()
   D. None of these

12.4 SUMMARY

- The built-in functions in Python are those functions whose tasks are already pre-defined in Python.
- These functions can be used directly in any program as deemed necessary by the programmers.
- Several built-in functions are present in the time module that are used to carry out different operations using time.
- Epoch in Python corresponds to the measurement of time starting from January 1, 1970.
- The time.time() function returns the number of seconds passed after epoch.
- The function, time.localtime(), is used to convert the number of seconds elapsed since epoch and returns the value in local time using the function struct_time().
- The asctime() function is used to convert the tuple returned to string form.
- sleep() is a convenient way of halting a program for specific needs for the desired period.
- By creating a date object means an object which represents a day, month and year.
- Comparison is done between the elements of the same group.
- The calendar module is an in-built module that includes all the functions and classes that take care of the operations related to a calendar.
- A list of indexes is returned for the days of the week by iterweekdays() method.
- Iterator of all the days of the month is returned by the use of itermonthdates() method.
- itermonthdays() method returns the day numbers and is similar to the itermonthdates method.
- monthdatescalendar() methods return full weeks in a month.
- monthdayscalendar() returns a list of full weeks and each week containing the list of days of a month.
- TextCalendar class is used to generate text calendars.
- HTMLCalendar class generates an HTML calendar.
- abs() function is used to present the absolute value of the given number.
- The bin() function converts the decimal number into a binary number.
- Interconversion of data types in Python can be done by using the bytes() function.

12.5 GLOSSARY

- **Float:** A floating-point number type
- **Complex:** A complex number type in Python.
- **Boolean Value:** Either True or False
- **Comparison:** checking if two strings are the same or not.
- **String:** Arrays representing Unicode characters.
- **Operator:** special symbols that carry out arithmetic or logical operations in python.
- **Argument:** Value passed to the function when it is called.
- **Function:** A set of actions that returns something to the programmer when called.
- **List:** An ordered collection of items that can be changed.
- **Tuple:** An ordered collection of items that cannot be changed.

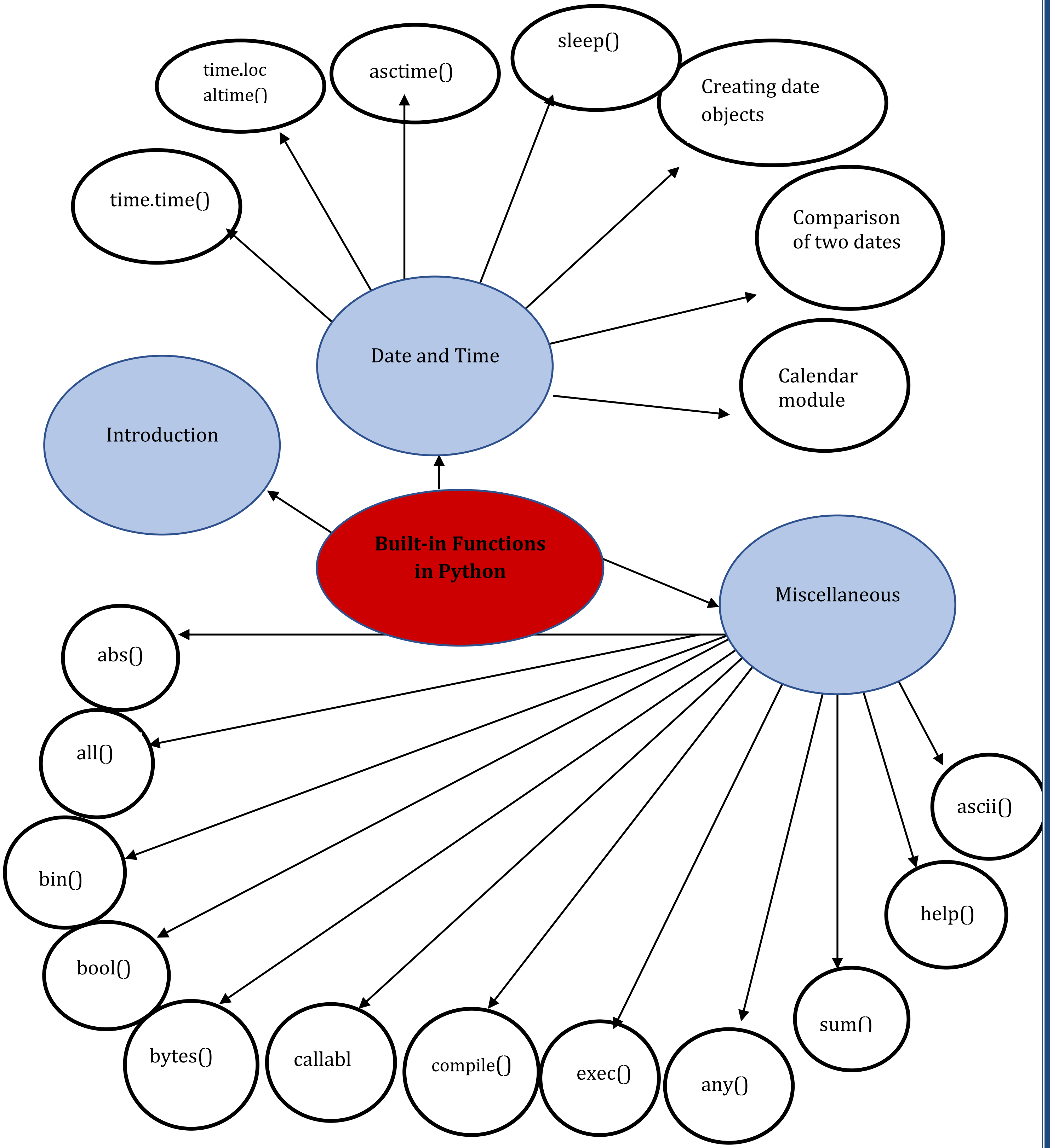

Fig 1: Conceptual fig

12.6 CASE STUDY

Deliver and collect packages swiftly and with a hassle-free app

Package Delivery App was made to ease the process of shipping and receiving parcels. The delivery app was developed to ensure smooth tracking of parcel till delivery. One can create a new shipment and track it until it is delivered. The UX designs were unique and still up to market standards.

The first challenge was to take up all the shipping information to start the delivery. The tracking screen and tracking screen shows detailed information with a presentable layout.

The app was built to be user-friendly for people belonging to different demographics so that they can navigate through the app easily. The track order screen keeps the entire history in one place.

Even the graphic designs were made keeping the company's brand in mind.
This app has a unique chat and review feature which makes it popular among the growing number of users.

***Source**: www.inianic.com*

Questions:

1. Discuss the functions used in completing the tracking and payment screens.
2. Create a wireframe for this app.

12.7 TERMINAL QUESTIONS

SHORT ANSWER QUESTIONS

Q1) What is an epoch?
Q2) What does iterweekdays() methods returns?
Q3) Create a Python program to get the date of the last Tuesday.
Q4) Create a Python program to get the week number.
Q5) Create a program to print yesterday, tomorrow, and today.

LONG ANSWER QUESTIONS

Q1) Create a program to get the current time.
Q2) Write a program to check if it's a leap year.
Q3) Create a program to demonstrate the use of any() function.
Q4) Create a program to demonstrate the use of the built-in function bool().
Q5) Create a program including bytes() including string.

ANSWERS

SELF ASSESSMENT QUESTIONS

1) pre-defined
2) time.localtime()
3) TRUE
4) TRUE
5) D. All of these
6) asctime()
7) bin()
8) FALSE
9) TRUE
10) C. minimum()

TERMINAL QUESTIONS:

SHORT ANSWER QUESTIONS

Answer 1:
Epoch in Python corresponds to the measurement of time starting from January 1, 1970.

Answer 2:
A list of indexes is returned for the days of the week by iterweekdays() method.

Answer 3:

```
from datetime import date
from datetime import timedelta
today = date.today()
x = (today.weekday() - 1) % 7
last_tuesday = today - timedelta(days = x)
print(last_tuesday)
```

Answer 4:

```
import datetime
print(datetime.date(2021, 3, 15).isocalendar()[1])
```

Answer 5:

```
import datetime
today = datetime.date.today()
yesterday = today - datetime.timedelta(days = 1)
tomorrow = today + datetime.timedelta(days = 1)
print('Yesterday was : ',yesterday)
print('Today is : ',today)
print('Tomorrow will be : ',tomorrow)
```

LONG ANSWER QUESTIONS:

Answer 1:

```
import datetime
print(datetime.datetime.now().time())
```

Answer 2:

```
def leap(y):
   if y % 400 == 0:
      return True
   if y % 100 == 0:
      return False
   if y % 4 == 0:
      return True
   else:
      return False
print(leap(1921))
print(leap(2004))
```

Answer 3:

```
l = [1, 2, 3, 0]
print(any(l))

l = [0, True]
print(any(l))
```

```
l = [0, False, 4]
print(any(l))

l = []
print(any(l))
```

Answer 4:

```
x = False
print(bool(x))

x = True
print(bool(x))

x = 4
y = 5
print(bool(x==y))

x = None
print(bool(x))

x = ()
print(bool(x))

x = {}
print(bool(x))

x = 0.0
print(bool(x))

x = 'Python'
print(bool(x))
```

Answer 5:

```
str1 = 'HelloWorld'

print ("Byte conversion with ignore error : " +
    str(bytes(str1, 'ascii', errors = 'ignore')))

print ("Byte conversion with replace error : " +
    str(bytes(str1, 'ascii', errors = 'replace')))

print ("Byte conversion with strict error : " +
    str(bytes(str1, 'ascii', errors = 'strict')))
```

12.8 SUGGESTED BOOKS AND REFERENCES

BOOKS:

1. Barry, Paul (2016), Head First Python: A Brain-Friendly Guide. Pearson Publishing.
2. Martin C. Brown (2018), Python: The Complete Reference. Packt Publishing
3. Allen Downey (2015), Learning with Python. 2nd Edition
4. Swaroop, C. H., (2013), A Byte of Python. Springer

E-REFERENCES:

- Python Built-in functions, last viewed on April 2, 2021 < https://www.w3schools.com/python/python_ref_functions.asp >
- Python Built-in functions with syntaxs and exmaples, last viewed on April 2, 2021 < https://www.w3schools.com/python/python_functions.asp >
- Python programming and function argument, last viewed on March 29, 2021 https://www.programiz.com/python-programming/function-argument \
- Python functions, last viewed on March 29, 2021 https://www.w3schools.com/python/python_functions.asp