SAMSUNG

Samsung Innovation Campus

Coding, Programming & Data Science

Together for Tomorrow!
Enabling People
Education for Future Generations

Chapter 7.

Data Processing, Descriptive Statistics, and Data Visualization

Coding, Programming & Data Science

Chapter Description

Chapter objectives

- ✓ Learners will be able to collect various types of large amounts of data and organize them in a form that can be analyzed.
- ✓ Learners will be able to generate various descriptive statistics for organized data using Pandas.
- ✓ Learners can visualize data using the Python Visualization Library.

Chapter contents

Unit 34.

Using Python Modules

Learning objectives

- ✓ Learners will be able to explain why modules are grouped into classes, functions, variables, execution codes, etc., and why they are needed.
- ✓ Learners will be able to look at documents on how to use the module and decrypt how to separate and use the functions, classes, and parameters, etc. that the module contains.
- ✓ Learners can select and use the import, from, and as statements in modules appropriately according to need.
- ✓ Learners will be able to generate as many integers and real data as they want using the random function when they encounter situations where unspecified test data is needed.
- ✓ Learners will be able to convert and generate values for a specific date and time using a data module.

Unit learning objective (2/3)

Learning overview

- ✓ Be able to use Python's standard module and external module.
- ✓ Be able to use the most commonly used standard library among many standard modules.
- ✓ Be able to create the necessary modules yourself and learn how to use them in other codes.
- ✓ Be able to bring up the entire module, classes, for functions in the module, and learn how to use it in your code.
- ✓ Although we haven't covered the concept of time series yet, be able to generate date and time data using datetime.

Concepts you will need to know from previous units

- ✓ Know how to use Python Functions.
- ✓ Know how to use Python class.

Keywords

- import
- Standard Module
- External Module
- random
- pip install
- datetime

Mission

1. Make a Dice Game!

What do we need to know about making a dice game?

- Today, we are going to make a dice game in which the computer and the user take turns to roll a fixed number of dice and see who gets the higher total. Each of them will get one chance to roll again and can decide which of the dices will be held or rerolled.
- The dice game problem involves a concept of randomness and an element of chance. Instead of deciding on an item, we will let the computer to pick something at random.
- Through this mission, we will learn how to make our own Python functions, which will allow us to name a block of code and then reuse it later by calling the name.

2. Your Dice Game will look like this!

※ To view the video clip, put the mouse on the box above, and the play button appears. Click it to watch.

Key concept

1. Modules

1.1. What is a Module?

- A module enables the Python code to be logically grouped, managed, and used. Normally, one Python .py file becomes one module. Functions, classes, or variables may be defined in the module, and may include an execution code.
- Simply put, it is a code file.
- It is divided into a standard module and an external module. The standard module refers to what is basically built into Python. In addition, modules created by other 3rd parties are called external modules.
  - Standard Module: Built-in modules within Python
  - External Module: Other modules made by a 3rd party.

1. Modules

1.1. What is a Module?

To use these modules, the module may be imported and used, and the import statement may invoke one or more modules within the code as shown below.

```
1 import Module Name
```

Line 1

- In case only one module is brought in for use.

```
1 import Module Name1, [Module Name2, Module Name3, …]
```

Line 1

- For cases of multiple modules.

1. Modules

1.1. What is a Module?

After importing a module using ‘import’, use the following method when using a specific function or variable of the module.

- ModuleName.Variable
- ModuleName.FunctionName()
- ModuleName.Class()

1. Modules

1.2. Standard Module, Standard Library

Standard modules are installed when installing Python. There is no need to memorize what is in the standard module, as it can be searched through a search engine or through Python's standard library at https://docs.python.org/3/library/.

Typical Functions of Standard Libraries

- Date and Time Modules
- Numbers and Math Modules
- File System Modules
- Operating System Modules
- Reading and Writing Data Format Modules such as HTML, XML, and JSON
- Internet Protocol Modules such as HTTP, SMTP, and FTP
- Multimedia Data Modules such as sound and video
- Localized Information Modules such as calls and dates

1. Modules

1.2. Standard Module, Standard Library

- Since the standard library is built-in, it does not require a separate installation process and can be used immediately by simply importing.

```
1 import math
```

Line 1

- A math module, one of the standard libraries with math-related functions

```
1 # It is used in the form of ModuleName.FunctionName()
2 math. sin(1)
```

```
0.8414709848078965
```

Line 1~2

- 1: It is used in the form of ModuleName.FunctionName()
- 2: An example of using a sin(x) function that obtains the sine value among several functions of the math module.

- To find out what other functions the Math module has, such as math.sin(x), visit https://docs.python.org/3/library/math.html. In this way, you can learn its basic use and use what you need through searching.

1. Modules

1.3. External Module, External Library

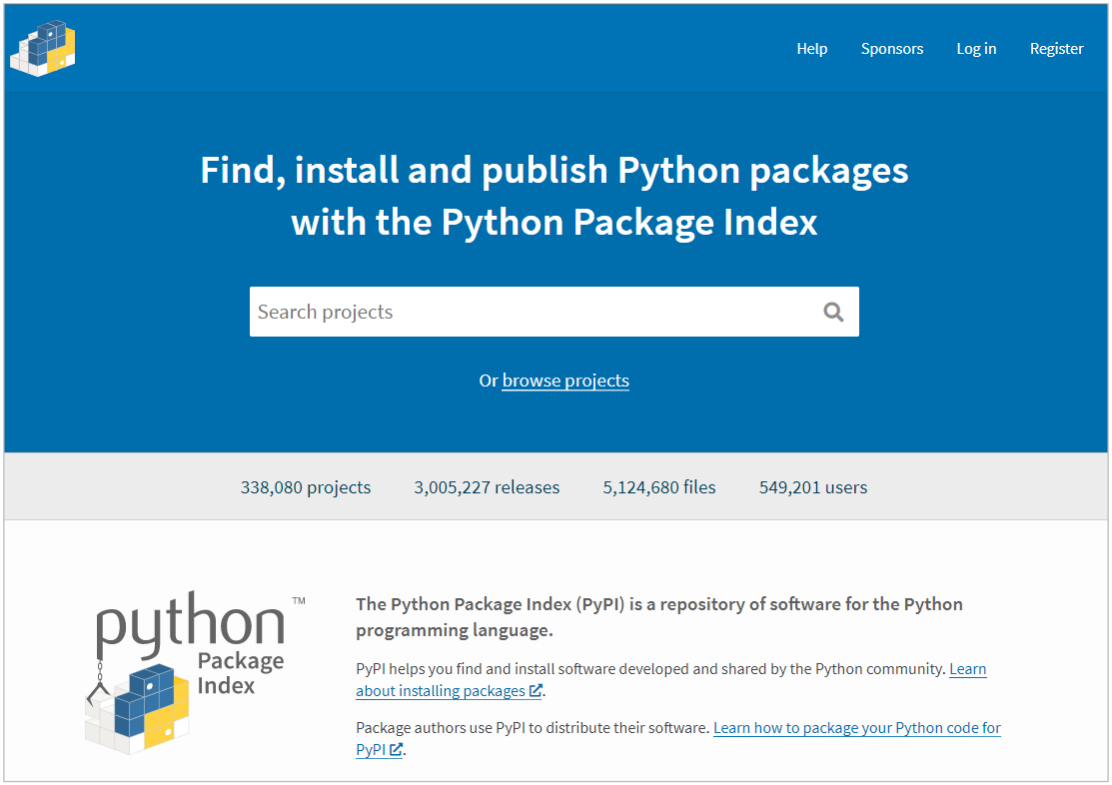
- Since external modules were created by other people (such as open-source libraries), the process of installing modules is required in order to use them in code.
- The most recommended and safe way to install an external library is to use pip. After Python 3.4, it is basically included in the Python binary installation program and can be easily used.
- Pip is a utility that allows for access of a widely used Python package index called PyPI(Python Package Index).

Ex If you want to install a module that has a special function, you can use the pip to install the available candidate library after searching in PyPI.

1. Modules

1.3. External Module, External Library

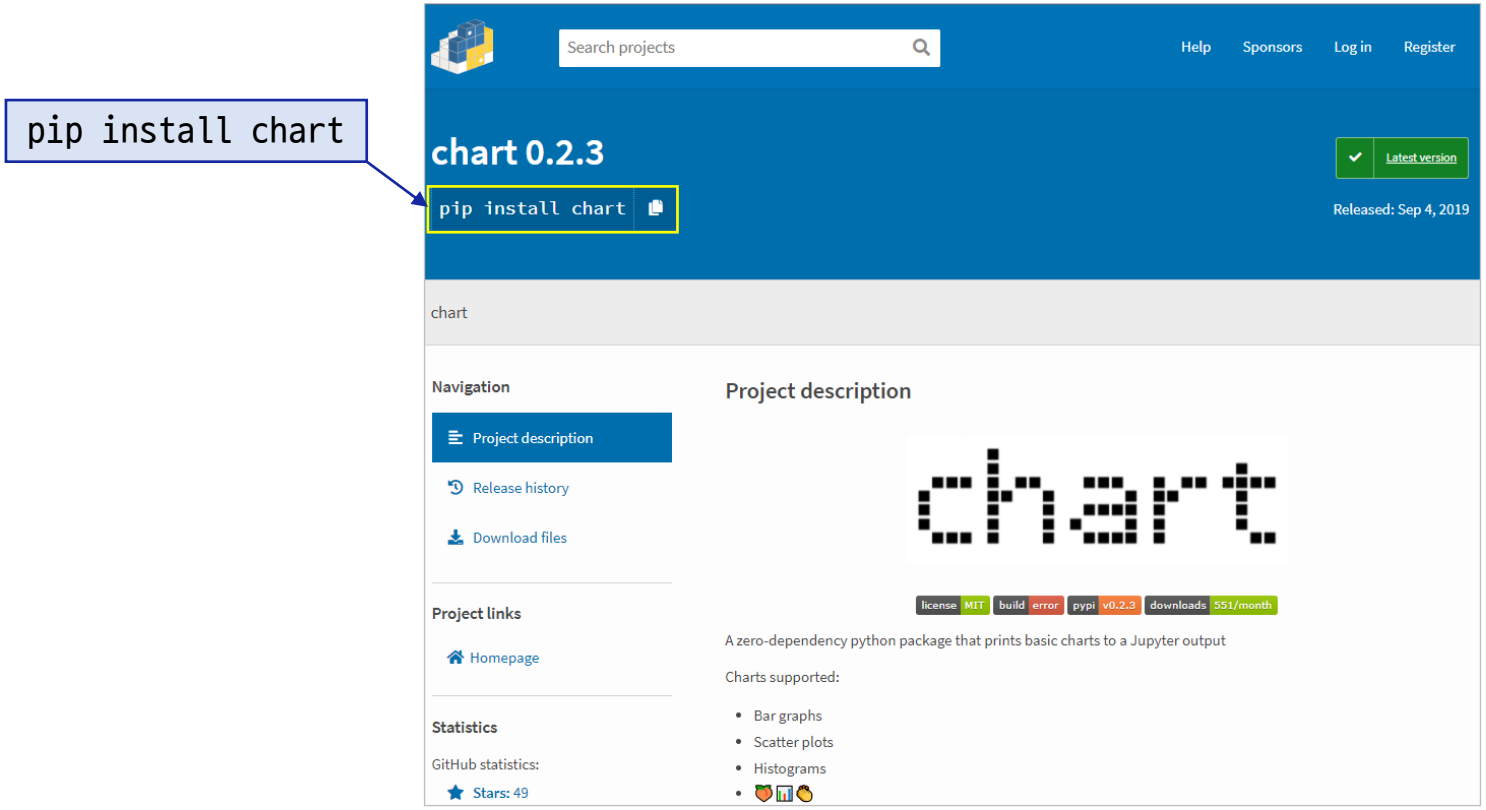
- Visit https://pypi.org/ in order to search the necessary functions.



1. Modules

1.3. External Module, External Library

- Examine the detail page in the library and copy the pip install under the module name if it is the desired function.



1. Modules

1.3. External Module, External Library

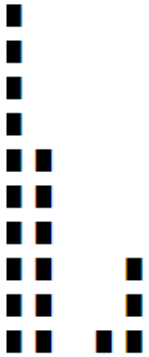
- Run Anaconda prompt and install the library after moving to the virtual environment you are currently using. The library installation instruction used as an example is a pip install chart.

```
(sic) PS C:₩Users₩User> pip install chart
Collecting chart
  Downloading chart-0.2.3.tar.gz (5.5 kB)
Building wheels for collected packages: chart
  Building wheel for chart (setup.py) ... done
  Created wheel for chart: filename=chart-0.2.3-py3-none-any.whl size=6950 sha256=b3bdfb412e18b8102c607fbbcfbd3a5f648083
4189951c7509cf2db2706339d9
  Stored in directory: c:₩users₩user₩appdata₩local₩pip₩cache₩wheels₩0a₩aa₩a2₩e1c444a91336651c0d310cdbe63a718edb2f79b3b79
3ac4d2a
Successfully built chart
Installing collected packages: chart
Successfully installed chart-0.2.3
(sic) PS C:₩Users₩User>
```

1. Modules

1.3. External Module, External Library

```
1 import chart
2 x = [1, 2, 4, 3, 3, 1, 7, 9, 9, 1, 3, 2, 1, 2]
3 chart.histogram(x)
```



Line 1, 3

- 1: Import the external library you installed through the import command.
- 3: Use histogram(x), which outputs a histogram chart (one of the chart library functions).

1. Modules

1.3. External Module, External Library

If you run the code and get an error message like the one below, read the message carefully. It is a message showing that the module cannot be found, and in this case, it is an error caused by the library not being installed. This is a mistake that is made more often at the beginner level than expected, and there are cases where a library is not installed in the virtual environment.

```
1 import chart
2 x = [1, 2, 4, 3, 3, 1, 7, 9, 9, 1, 3, 2, 1, 2]
3 chart.histogram(x)
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
~₩AppData₩Local₩Temp/ipykernel_7444/2810548644.py in <module>
----> 1 import chart
      2 x = [1, 2, 4, 3, 3, 1, 7, 9, 9, 1, 3, 2, 1, 2]
      3 chart.histogram(x)

ModuleNotFoundError: No module named 'chart'
```

Let's practice searching, installing, and using a library that outputs emoticons.

1. Modules

1.4. Make Your Own Module and Bring It Up For Use

In addition to being able to import and use the Python module .py file, the entire script in the module file can be executed immediately. Let's practice bringing up modules through the practice of calculating the hospital funds.

We sell event tickets to raise funds for hospitals.

Each individual participating in the event has to pay 5t +3.

T is the number of tickets purchased by one person.

1. Modules

1.4. Make Your Own Module and Bring It Up For Use

```
1  # Save the name of this file as fund_cal.py.
2
3  def fund(t):
4      return t5+3
```

Line 1

- Save the name of this file as fund_cal.py.

1. Modules

1.4. Make Your Own Module and Bring It Up For Use

```
1  import donate
2
3  def main():
4      t = int(input("Enter the total number of people who participated in the hospital donation event"))
5      total = fund_cal.fund(t)
6
7      print("Total Donation Amount :", total)
8
9  main()
```

Line 1

- The plan is to bring and use the donate.py file, a module created and stored above. Be careful not to write the extension for the filename.py.

1. Modules

1.4. Make Your Own Module and Bring It Up For Use

Ex An example of when an external file (*.py) is retrieved and the module selects and calls only specific functions.

```
1  # Store this file under the name cal_n.py
2
3  def sum_n(n):
4      return n * (n+1) // 2
5
6  def sum_n2(n):
7      t = 0
8      for i in range(1, n+1):
9          t = t + i
10     return t
```

Line 1, 3, 4, 6, 12, 13

- 1: Store this file under the name cal_n.py.
- 3: An algorithm that calculates the value obtained by adding all the numbers from 3:1 to the input n.
- 4: Two slashes are divided into two
- 6: An algorithm that adds all the consecutive numbers from 6:1 input n

1. Modules

1.4. Make Your Own Module and Bring It Up For Use

Ex An example of when an external file (*.py) is retrieved and the module selects and calls only specific functions.

```
1  # Store this file under the name cal_n.py
2
3  def sum_n(n):
4      return n * (n+1) // 2
5
6  def sum_n2(n):
7      t = 0
8      for i in range(1, n+1):
9          t = t + i
10     return t
```

Line 1, 3, 4, 6, 12, 13

- 12: In order to check whether the algorithm above works well, verification is performed using print(sum_n(10)) function. However, this example code was annotated as an unnecessary code area because it was for the practice of bringing up the module.
- 13: print(sum_n2(100))

1. Modules

1.4. Make Your Own Module and Bring It Up For Use

```
1  import cal_n
2
3  def main():
4      n = int(input("Enter a desired number for calculation : "))
5
6      sum_v = cal_n.sum_n(n)
7      print("The sum of adding from 1 until", n, "is: ", sum_v)
8
9      sum_vv =cal_n.sum_n2(n)
10     print("The sum of adding consecutively from 1 until", n, "is: ", sum_vv)
11
12 main()
```

Line 1

- Note that you do not use the .py extension when importing a file into a module.

1. Modules

1.5. Python Syntax: From

The module contains many variables and functions, all of which are extremely rare to find and use 100%. When you want to use only a specific function in the module, you can use the 'From' syntax in the following format.

- from Module import FunctionName
- from Module import ClassName
- from Module import VariableName

1. Modules

1.5. Python Syntax: From

If we apply it to the math.sin(1) we practiced earlier, it looks like the following.

```
1 from math import sin
2 sin(1)
```

```
0.8414709848078965
```

Line 2

- We can use it by only using the function name, as opposed to adding the module name ‘math’ in the front.

1. Modules

1.5. Python Syntax: From

Several variables or functions that you want to use in the module can also be used in the following format.

```
1 from math import sin, cos, tan
2 sin(1)
```

```
0.8414709848078965
```

Line 1

- Several functions names can be called at once using ‘ , ’.

```
1 cos(1)
```

```
0.5403023058681398
```

```
1 tan(1)
```

```
1.5574077246549023
```

1. Modules

1.5. Python Syntax: From

However, if it is inconvenient to use the module name in front of it, then you can code using only the function name. The entire function of the module can be brought using the form 'from ModuleName import*'

```
from math import *
```

```
1 sin(1)
```

```
0.8414709848078965
```

Line 1

- Even though the function sin is not written after import, it can be used only with the function name.

1. Modules

1.5. Python Syntax: From

```
1 floor(3.2)
```

```
3
```

Line 1

- You can round down without writing the function ‘floor’ after import.

```
1 ceil(4.6)
```

```
5
```

Line 1

- You can round up without writing the function ‘ceil’ after import.

1. Modules

1.6. Python Syntax: As

The name of the module is long, so it is sometimes cumbersome to write the code. And sometimes the names overlap when installing and using an external library. In this case, change the name of the library using the as syntax. It can be used as a short word.

```
import ModuleName as DesiredModuleName(Identifier)
```

```
1 import math as m
```

```
1 m.sin(1)
```

```
0.8414709848078965
```

```
1 m.floor(2.7)
```

```
2
```

1. Modules

1.6. Python Syntax: As

When describing 'from', it was explained that various variables or function names can be retrieved at once using ' , ' when importing,

```
from Module import Variable as Name1, Function as Name2, Class as Name3
```

```
1 from math import sin, cos, tan
```

Line 1

- It can be called one after another using 'as' even while changing it using abbreviations.

```
1 from math import sin as s, cos as c
```

```
1 s(1)
```

```
0.8414709848078965
```

```
1 c(1)
```

```
0.5403023058681398
```

2. Using Typical Standard Libraries

2.1. Using the Random Module to Create Random Numbers

- This module is used to make random numbers. And it can be used to sample and extract some parts from the list.
- Random means that the results appear unpredictably every time.

  Ex When you roll a dice, one number is selected randomly from 1 to 6.

2. Using Typical Standard Libraries

2.1. Using the Random Module to Create Random Numbers

1) Example of Generating Random Numbers

- random() is a function belonging to the random module which randomly generates floats between 0 and 1.

```
1 import random
2 i = 0
3 for i in range(10):
4     a = random.random()
5     print(a)
```

```
0.901282928943991
0.3097445865295111
0.5379300227358534
0.39417771664028634
0.8583507092624373
0.17331976492203316
0.5986237853706852
0.6502718258390799
0.5832678067472766
0.3299098329677781
```

Line 4, 5

- 4: random() randomly generates floats among numbers greater than or equal to 0 and less than 1.
- 5: Each time it is executed, a number between 0 and 1 is randomly returned. Another number returns randomly when 10 loops are executed.

2. Using Typical Standard Libraries

2.1. Using the Random Module to Create Random Numbers

2) An example of sampling a part of a list or a tuple and extracting randomly

- sample (list name, number of samples) is a function that randomly extracts as many samples as the number of samples from the list. It is used for random sampling without duplication.
- Instead of the list, a tuple and set may also be used as a collection of data extraction targets.

```
1  import random
2
3  data = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]
4  a = random.sample(data,6)
5
6  print (a)
```

```
[13, 12, 3, 10]
```

Line 4

- We randomly extract the six samples from a collection called ‘data’ and store them in the variable a.

2. Using Typical Standard Libraries

2.1. Using the Random Module to Create Random Numbers

2) An example of sampling a part of a list or a tuple and extracting randomly

- The choice() function randomly extracts any element from the collection without specifying the number of samples.

```
1  import random
2
3  dog_list = ('Labrador Retriever', 'German Shepherd', 'Bulldog', 'Beagle', 'Yorkshire Terrier' )
4  my_lovely_dog = random.choice(dog_list)
5
6  print(my_lovely_dog)
```

```
Beagle
```

Line 3, 4

- 3: Tuple collection made of dog names
- 4: Random elements are extracted and stored in the variable my_lovely_dog.

2. Using Typical Standard Libraries

2.1. Using the Random Module to Create Random Numbers

random.random()

Return the next arbitrary floating-point number in the section [0.0, 1.0]

```
1 random.random()
```

```
0.8198361770392332
```

2. Using Typical Standard Libraries

2.2. Functions of the Random Module that Generates an Integer Distribution

random.uniform(a,b)

- Returns an arbitrary float N that satisfies the condition if a <= b, then a <= N <= b, and if b < a, then b <= N <= a.
- The termination value b may or may not be included in the range according to the float position of a + (b-a) * random().
- Simply put, you can set it to return any float in the range where a is the minimum value and b is the maximum value.

```
1  import random
2
3  x = random.uniform(1.5,10)
4  y = random.uniform(10,1.5)
5
6  print("If a <=b, then a <= N <=b,", x)
7  print("If b < a, then b <=N <=a", y)
8  print("Set to return any float in the range where a is the minimum value and b is the maximum value.")
```

```
a <= b 일 때 a <= N <= b, 2.3489833131101863
b < a 일 때 b <= N <= a 2.216341666354145
Set to return any float in the range where a is the minimum value and b is the maximum value.
```

2. Using Typical Standard Libraries

2.3. Functions of the Random Module that Generates an Integer Distribution

random.randint(a, b)

Returns any integer (a < N < b) between a and b as the minimum and maximum value, respectively.

```
1  import random
2
3  x = random.randint(1,100)
4  print(x)
```

```
36
```

Line 3

- Returns an integer between 1 and 100 and stores it in variable x.

2. Using Typical Standard Libraries

2.3. Functions of the Random Module that Generates an Integer Distribution

random.randrange(start, stop, step)

Returns an arbitrary integer as a step from the start value to the stop value.

```
1  import random
2
3  x = random.randrange(1,100,2)
4  print(x)
```

```
67
```

2. Using Typical Standard Libraries

2.3. Functions of the Random Module that Generates an Integer Distribution

random.randint(a, b)

Returns any integer N that satisfies a <= N <= b.

```
1  import random
2
3  roll = random.randint(1, 10)
4  print(f'You rolled {roll}.')
```

```
You rolled 5.
```

TIP

- A module that generates random numbers should not be used for security purposes. For security or encryption, it is recommended to use the secrets module.

2. Using Typical Standard Libraries

2.4. Time Module

time.sleep(secs)

- It is the most commonly used function among time module functions.
- It functions to pause the execution of the thread called for a given second.

```
1  import time
2
3  for i in range(10):
4      print("Hello", i+1)
5      time.sleep(2)
```

```
Hello 1
Hello 2
Hello 3
Hello 4
Hello 5
Hello 6
Hello 7
Hello 8
Hello 9
Hello 10
```

Line 5

- All executions are paused for 2 seconds at this part while the loop is in execution.

2. Using Typical Standard Libraries

2.4. Time Module

time.sleep(secs)

```
1 import time
2
3 a= time.time()
4
5 print(a)
6 print(b)
```

```
1629940763.8416274
Thu Aug 26 10:19:23 2021
```

Line 3, 4

- 3: It is a function of finding the current time. As of 0h 0m 0s on January 1st, 1970, it informs you of the past time in seconds. However, the return value is returned to a real value that is difficult to read.
- 4: It is a function for returning the form of time that we can understand.

Unix timestamp

- It is also called Epoch time. The elapsed time from 00:00:00(UTC) on January 1st, 1970 is converted into seconds and expressed as an integer.

2. Using Typical Standard Libraries

2.4. Time Module

time.sleep(secs)

Let's make a simple electronic watch.

```
1  import time
2
3  for i in range(10):
4      time.sleep(1)
5      print(time.ctime())
```

```
Thu Aug 26 10:25:12 2021
Thu Aug 26 10:25:13 2021
Thu Aug 26 10:25:14 2021
Thu Aug 26 10:25:15 2021
Thu Aug 26 10:25:16 2021
Thu Aug 26 10:25:17 2021
Thu Aug 26 10:25:18 2021
Thu Aug 26 10:25:19 2021
Thu Aug 26 10:25:20 2021
Thu Aug 26 10:25:21 2021
```

2. Using Typical Standard Libraries

2.4. Time Module

time.strftime('Format', Time object)

When the %y format code is provided, a two-digit year can be parsed. Values 69 to 99 are mapped from 1969 to 1999, and values 0 to 68 are mapped from 2000 to 2068.

%a : Name of the Week

%b : Name of the Month

%d : Day of the Month in Decimal

%Y : Year in Decimal

- https://docs.python.org/3/library/time.html?highlight=time#time.strftime

2. Using Typical Standard Libraries

2.4. Time Module

time.strftime('Format', Time object)

```
1 from time import gmtime, strftime
2 strftime("%a, %d %b %Y %H:%M:%S +0000", gmtime())
```

```
'Thu, 26 Aug 2021 01:15:47 +0000'
```

```
1 from time import localtime, strftime
2 strftime("%a, %d %b %Y %H:%M:%S +0000", localtime())
```

```
'Thu, 26 Aug 2021 13:45:52 +0000'
```

2. Using Typical Standard Libraries

2.5. Using the Basics of the Datetime Moduletime Module

As a module related to date and time, it is often used to create date formats. Below is a summary of various cases that print out basic dates.

datetime.date.today()

- Returns the current date.

```
1  import datetime
2
3  today = datetime.date.today()
4
5  print(today)
6  print(today.year, "year")
7  print(today.month, "month")
8  print(today.day, "day")
```

```
2021-08-26
2021 year
8 month
26 day
```

Line 3, 6

- 3: Returns the current date to the variable today and stores it
- 6: Separates the information about year

2. Using Typical Standard Libraries

2.5. Using the Basics of the Datetime Moduletime Module

As a module related to date and time, it is often used to create date formats. Below is a summary of various cases that print out basic dates.

datetime.date.today()

- Returns the current date.

```
1  import datetime
2
3  today = datetime.date.today()
4
5  print(today)
6  print(today.year, "year")
7  print(today.month, "month")
8  print(today.day, "day")
```

```
2021-08-26
2021 year
8 month
26 day
```

Line 7, 8

- 7: Separates the information about month
- 8: Separates the information about date

2. Using Typical Standard Libraries

2.5. Using the Basics of the Datetime Moduletime Module

datetime.datetime.now()

- Returns the current date up to hours, minutes, and seconds.

```
1  import datetime
2
3  now= datetime.datetime.now()
4
5  print(now)
6  print(now.year, "year")
7  print(now.month, "month")
8  print(now.day, "day")
9  print(now.hour, "hour")
10 print(now.minute, "minute")
11 print(now.second, "second")
```

```
2021-08-26 13:58:29.616642
2021 year
8 month
26 day
13 hour
58 minute
29 second
```

2. Using Typical Standard Libraries

2.5. Using the Basics of the Datetime Moduletime Module

```
1  import datetime
2
3  now= datetime.datetime.now()
4
5  print(now)
6  print(now.year, "year")
7  print(now.month, "month")
8  print(now.day, "day")
9  print(now.hour, "hour")
10 print(now.minute, "minute")
11 print(now.second, "second")
```

Line 3, 6, 7

- 3: Year, month, day, hour, minute, second
- 6: Separates the information about year
- 7: Separates the information about month

2. Using Typical Standard Libraries

2.5. Using the Basics of the Datetime Moduletime Module

```
1  import datetime
2
3  now= datetime.datetime.now()
4
5  print(now)
6  print(now.year, "year")
7  print(now.month, "month")
8  print(now.day, "day")
9  print(now.hour, "hour")
10 print(now.minute, "minute")
11 print(now.second, "second")
```

Line 8, 9, 10, 11

- 8: Separates the information about day
- 9: Separates the information about hour
- 10: Separates the information about minute
- 11: Separates the information about second

2. Using Typical Standard Libraries

2.6. Using datetime.timedelta to Indicate the Difference Between Two Dates and Time

```
datetime.timedelta(days=0, seconds=0, microseconds=0, milliseconds=0, minutes=0, hours=0, weeks=0)
```

All factors are optional, not essentials, and the default value is zero. You can enter it as an integer or float. You can use both positive and negative numbers.

- Milliseconds are converted into 1000 microseconds.
- Minutes are converted into 60 seconds.
- Hours are converted into 3600 seconds.
- Weeks are converted into 7 days.

2. Using Typical Standard Libraries

2.6. Using datetime.timedelta to Indicate the Difference Between Two Dates and Time

```
datetime.timedelta(days=0, seconds=0, microseconds=0, milliseconds=0, minutes=0, hours=0, weeks=0)
```

```
1  from datetime import timedelta
2
3  delta = timedelta(
4      days=50,
5      seconds=27,
6      microseconds=10,
7      milliseconds=29000,
8      minutes=5,
9      hours=8,
10     weeks=2)
11
12 print(delta)
```

```
64 days, 8:05:56.000010
```

2. Using Typical Standard Libraries

2.6. Using datetime.timedelta to Indicate the Difference Between Two Dates and Time

datetime.timedelta(days=0, seconds=0, microseconds=0, milliseconds=0, minutes=0, hours=0, weeks=0)

Let's find the date that is 30 days from May 3rd, 2000.

```
1  from datetime import timedelta
2
3  d = datetime.datetime(2000, 5, 3)
4
5  delta = datetime.timedelta(days = 30)
6
7  print(d + delta)
```

```
2000-06-02 00:00:00
```

Line 3

- Objects can be created by adding years, months, days, hours, minutes, seconds, and microseconds to datetime.datetime.

2. Using Typical Standard Libraries

2.7. Using Replace to Replace Elements of a Specific Time

```
datetime.replace(year=, month= , day= , hour=, minute=, second=, microsecond= )
```

- You can change the elements of a specific time.

```
1  from datetime import datetime
2
3  now = datetime.now()
4
5  print(now)
6
7  replace_time = now.replace(month = 12, day = 30)
8
9  print(replace_time)
```

```
2021-08-26 15:18:41.392983
2021-12-30 15:18:41.392983
```

Line 7

- Change the month to December and the day to the 30th.

2. Using Typical Standard Libraries

2.8. OS Modules with Functions Related to the Operating System

It is a module that has functions related to the operating system. You can create a new folder on our computer or look at the list of files inside the folder using the OS module.

- **os.mkdir(”Folder Name”)**: Creates a folder.
- **os.rmdir(”Folder Name")**: Deletes the folder.
- **os.getcwd()**: Returns the current path.
- **os.listdir()**: Inquires the list of files and directories

2. Using Typical Standard Libraries

2.8. OS Modules with Functions Related to the Operating System

```
1 import os
2 os.getcwd()
```

```
'C:\\Users\\User\\Documents\\sic_project'
```

```
1 import os
2 os.listdir()
```

```
['.ipynb_checkpoints', '01.ipynb', 'data','UNIT34_Using Python Modules.ipynb']
```

Paper coding

Try to fully understand the basic concept before moving on to the next step.

Lack of understanding basic concepts will increase your burden in learning this course, which may make you fail the course.

It may be difficult now, but for successful completion of this course we suggest you to fully understand the concept and move on to the next step.

Q1. Randomly select three multiples of 5 from the range 0 to 100 using the random module and print them in the form of a list.

Write the entire code and the expected output results in the note.

Q2.. Use timedelta to create a program that prints a 100-day anniversary from a special day of yours. It doesn't have to be a 100-day anniversary, so feel free to make your own special anniversary calculator.

Write the entire code and the expected output results in the note.

Let’s code

Steps for Writing Python Code

STEP 1

- Ask the user to enter the number of dice. Then, convert the data type of the number of dice from string to integer.

```
# step1 in main program area - start game
number_dice = input('Enter number of dice:')
number_dice = int(number_dice)
```

STEP 2

- Ask the user to press any key to start the dice game.

```
ready = input('Ready to start? Hit any key to continue')
```

STEP 3

- We need to generate random numbers for rolling a dice. We're going to import and use the random module from the Python library. Write the line on the very top of the code.

```
import random
```

Steps for Writing Python Code

STEP 4

- Define a function named "roll_dice(n)." Make sure to create the function right after the import statement. It will use a parameter "n," which is the number of dice. We are going to call this function in the main program for rolling a dice. Within the roll_dice(n) function, create an empty list that will store the dice numbers rolled.

```
def roll_dice(n):
    dice = [] # start with empty list of dice
```

STEP 5

- Add random numbers that are between 1 and 6 to the list of dice. Create the numbers as much as the number of dice indicated as a parameter. When you're done with appending all the random numbers required to the list, return the list of dice.

```
def roll_dice(n):
    dice = [] # start with empty list of dice
    # add random numbers between 1 to 6 to the list
    for i in range(n):
        dice.append(random.randint(1,6))
    return dice
```

Steps for Writing Python Code

STEP 6

In the main programming area, call the roll_dice(n) function with number_dice as a parameter. Name the list of dice returned by the function as "user_rolls." Display the user's first roll on the screen.

```
# step 2 in main program area - roll dice
# User turn to roll
user_rolls = roll_dice(number_dice)
print('User first roll: ', user_rolls)
```

Steps for Writing Python Code

STEP 7

Get the user's choices for holding on or rerolling each dice. Check if the user enters the right number of choices that matches the number of dice. If the length of the user's input and the number of dice do not match, ask the user to re-enter the choices.

```
# step 3 - get user choices
user_choices = input("Enter - to hold or r to \
roll again :")
# check length of user input
while len(user_choices) != number_dice:
    print('You must enter', number_dice, \
    'choices')
    user_choices = input("Enter - to hold or r \
    to roll again :")
```

Steps for Writing Python Code

STEP 8

- We are going to use a sleep( ) function to pause the program for seconds. Import the time module from the Python library. Write the code below the statement for importing the random module.

```
import random

import time
```

STEP 9

- Define a function named "roll_again(choices, dice_list)," two lines after the function definition of roll_dice(n). It will use parameters of either the user or computer's choices and of the list of dices. Display a message, and pause the program for 3 seconds to wait for the computer to roll a dice. Refer to the side note on the next slide for using the sleep( ) function of the time module.

```
def roll_again(choices, dice_list):
    print('Rolling again ...')
    time.sleep(3)
```

Steps for Writing Python Code

STEP 10

Step through the choices, and if a character within the string is "r" (roll again), replace the dice at the index of the dice list to a new random number between 1 and 6. When you're done with the for loop, pause the program again for 3 seconds. Since the function is a void function, it does not return anything at the end.

```
def roll_again(choices, dice_list):
    print('Rolling again ...')
    time.sleep(3)
    for i in range(len(choices)):
        if choices[i] == 'r':
            dice_list[i] = random.randint(1,6)
    time.sleep(3)
```

Important

time.sleep(secs)

- pauses, stops, waits, or sleeps your Python program for secs. Here, secs is the number of seconds that the Python code should pause execution, and the argument should be either an integer or a float.

Steps for Writing Python Code

STEP 11

In the main programming area, call the function roll_again(choices, dice_list) with user_choices and user_rolls as parameters. The list of dice will be updated through the execution of the function, based on the user's choices. Display the user's new roll on the screen.

```
# step 4 - roll again based on user choices
roll_again(user_choices, user_rolls)
print('Player new Roll: ', user_rolls)
```

STEP 12

Now, it's the computer's turn to roll a dice. Call the function roll_dice(n) with number_dice as a parameter. Name the list of dice returned by the function as "computer_rolls." Display the computer's first roll on the screen.

```
# step 5 - computer's turn to roll
print('Computers turn ')
computer_rolls = roll_dice(number_dice)
print('Computer first roll: ', computer_rolls)
```

Steps for Writing Python Code

STEP 13

Define a function named "computer_strategy(n)," which will decide on the computer's choices for whether to hold on or to re-roll each dice. It will use a parameter "n," which is the number of dice. Display a message that the computer is thinking, and pause the program for 3 seconds. Also, create an empty list that will store the computer's choices on each dice.

```
def computer_strategy1(n):
    # create computer choices : roll everything again
    print('Computer is thinking ...')
    time.sleep(3)
    choices = '' # start with an empty list of choices
```

TIP

- You can create an empty list in three ways:
  ① Name of an empty list = [ ]
  ② Name of an empty list = list[ ]
  ③ Name of an empty list = ' '

Steps for Writing Python Code

STEP 14

Next, step through each element in the list of dice for the computer. If the dice in the computer's dice list is less than 5, append "r" (roll again) to the computer's choices list. If it is 5 or 6, append "-" (hold) to the computer's choices list. Lastly, return the list of choices that the computer has made.

```
def computer_strategy2(n):
    # create computer choices: roll if < 5
    print('Computer is thinking ...')
    time.sleep(3)
    choices = '' # start with an empty list of choices
    for i in range(n):
        if computer_rolls[i] < 5:
            choices = choices + 'r'
        else:
            choices = choices + '-'
    return choices
```

Steps for Writing Python Code

STEP 15

- In the main programming area, call the function computer_strategy(n) with number_dice as a parameter. The list of dice will be updated through the execution of the function, based on the user's choices. Display the user's new roll on the screen.

```
# step 6
# decide on what choice - using one of the strategy functions
computer_choices = computer_strategy2(number_dice)
print('Computer Choice: ', computer_choices)
```

STEP 16

- Now, it's the computer's turn to roll a dice. Call the function roll_dice(n) with number_dice as a parameter. Name the list of dice returned by the function as "computer_rolls." Display the computer's first roll on the screen.

```
# Computer rolls again using the choices it made
roll_again(computer_choices, computer_rolls)
print('Computer new Roll: ', computer_rolls)
```

Steps for Writing Python Code

STEP 17

Define a function named "find_winner(cdice_list, udice_list)," which will determine the winner for the dice game. It will use the lists of dices for each computer and user as parameters. With the Python's sum( ) function, calculate the totals of the dice numbers from each computer and user's list of dices. Then, display the totals on the screen.

```
def find_winner(cdice_list, udice_list):
    computer_total = sum(cdice_list)
    user_total = sum(udice_list)
    print('Computer total', computer_total)
    print('User total',user_total )
```

Important

sum(iterable, start)

- returns a number, the sum of all items in an iterable. Here, iterable is a required parameter, which is the sequence or a list of numbers to sum. On the other hand, start is an optional parameter, which is a value that is added to the return value.

Steps for Writing Python Code

STEP 18

If the user has a higher total, display that the user is a winner. If the computer has a higher total, display that the computer is a winner. Otherwise, the user and the computer have the same total, so display that it is a tie. Since the function is a void function, it does not return anything.

```
def find_winner(cdice_list, udice_list):
    computer_total = sum(cdice_list)
    user_total = sum(udice_list)
    print('Computer total', computer_total)
    print('User total',user_total )
    if user_total > computer_total:
        print('User wins')
    elif user_total < computer_total:
        print('Computer wins')
    else:
        print('It is a tie!')
```

Steps for Writing Python Code

STEP 19

In the main programming area, call the find_winner(cdice_list, udice_list) with computer_rolls and user_rolls as parameters. The parameters, which are the lists of dices, have been updated through the previous functions. The program will end by determining the winner of the dice game in the find_winner(cdice_list, udice_list) function.

```
# final line in code - deciding who wins
find_winner(computer_rolls,user_rolls)
```

Finalized Code for a Dice Game

```
import random

import time

def roll_dice(n):
    dice = [] # start with empty list of dice
    # add random numbers between 1 to 6 to the list
    for i in range(n):
        dice.append(random.randint(1,6))
    return dice
```

Finalized Code for a Dice Game

```
def find_winner(cdice_list, udice_list):
    computer_total = sum(cdice_list)
    user_total = sum(udice_list)
    print('Computer total', computer_total)
    print('User total',user_total )
    if user_total > computer_total:
        print('User wins')
    elif user_total < computer_total:
        print('Computer wins')
    else:
        print('It is a tie!')

def roll_again(choices, dice_list):
    print('Rolling again ...')
    time.sleep(3)
    for i in range(len(choices)):
        if choices[i] == 'r':
            dice_list[i] = random.randint(1,6)
    time.sleep(3)
```

Finalized Code for a Dice Game

```
def computer_strategy1(n):
    # create computer choices : roll everything again
    print('Computer is thinking ...')
    time.sleep(3)
    choices = '' # start with an empty list of choices
    for i in range(n):
        choices = choices + 'r'
    return choices

def computer_strategy2(n):
    # create computer choices: roll if < 5
    print('Computer is thinking ...')
    time.sleep(3)
    choices = '' # start with an empty list of choices
    for i in range(n):
        if computer_rolls[i] < 5:
            choices = choices + 'r'
        else:
            choices = choices + '-'
    return choices
```

Finalized Code for a Dice Game

```
# step1 in main program area - start game
number_dice = input('Enter number of dice:')
number_dice = int(number_dice)
ready = input('Ready to start? Hit any key to continue')


# step 2 in main program area - roll dice
# User turn to roll
user_rolls = roll_dice(number_dice)
print('User first roll: ', user_rolls)

# step 4 - get user choices
user_choices = input("Enter - to hold or r to \
roll again :")
# check length of user input
while len(user_choices) != number_dice:
    print('You must enter', number_dice, \
    'choices')
    user_choices = input("Enter - to hold or r \
    to roll again :")
```

Finalized Code for a Dice Game

```
# step 5 - roll again based on user choices
roll_again(user_choices, user_rolls)
print('Player new Roll: ', user_rolls)

# Computer's turn to roll
print('Computers turn ')
computer_rolls = roll_dice(number_dice)
print('Computer first roll: ', computer_rolls)

# step 6
# decide on what choice - using one of the strategy functions
computer_choices = computer_strategy2(number_dice)
print('Computer Choice: ', computer_choices)
# Computer rolls again using the choices it made
roll_again(computer_choices, computer_rolls)
print('Computer new Roll: ', computer_rolls)


# final line in code - deciding who wins
find_winner(computer_rolls,user_rolls)
```

Pair programming

Pair Programming Practice

Guideline, mechanisms & contingency plan

Preparing pair programming involves establishing guidelines and mechanisms to help students pair properly and to keep them paired. For example, students should take turns “driving the mouse.” Effective preparation requires contingency plans in case one partner is absent or decides not to participate for one reason or another. In these cases, it is important to make it clear that the active student will not be punished because the pairing did not work well.

Pairing similar, not necessarily equal, abilities as partners

Pair programming can be effective when students of similar, though not necessarily equal, abilities are paired as partners. Pairing mismatched students often can lead to unbalanced participation. Teachers must emphasize that pair programming is not a “divide-and-conquer” strategy, but rather a true collaborative effort in every endeavor for the entire project. Teachers should avoid pairing very weak students with very strong students.

Motivate students by offering extra incentives

Offering extra incentives can help motivate students to pair, especially with advanced students. Some teachers have found it helpful to require students to pair for only one or two assignments.

Pair Programming Practice

Prevent collaboration cheating

The challenge for the teacher is to find ways to assess individual outcomes, while leveraging the benefits of collaboration. How do you know whether a student learned or cheated? Experts recommend revisiting course design and assessment, as well as explicitly and concretely discussing with the students on behaviors that will be interpreted as cheating. Experts encourage teachers to make assignments meaningful to students and to explain the value of what students will learn by completing them.

Collaborative learning environment

A collaborative learning environment occurs anytime an instructor requires students to work together on learning activities. Collaborative learning environments can involve both formal and informal activities and may or may not include direct assessment. For example, pairs of students work on programming assignments; small groups of students discuss possible answers to a professor's question during lecture; and students work together outside of class to learn new concepts. Collaborative learning is distinct from projects where students "divide and conquer." When students divide the work, each is responsible for only part of the problem solving and there are very limited opportunities for working through problems with others. In collaborative environments, students are engaged in intellectual talk with each other.

Q1. Complete the creative artwork by looking at the document with your colleagues.

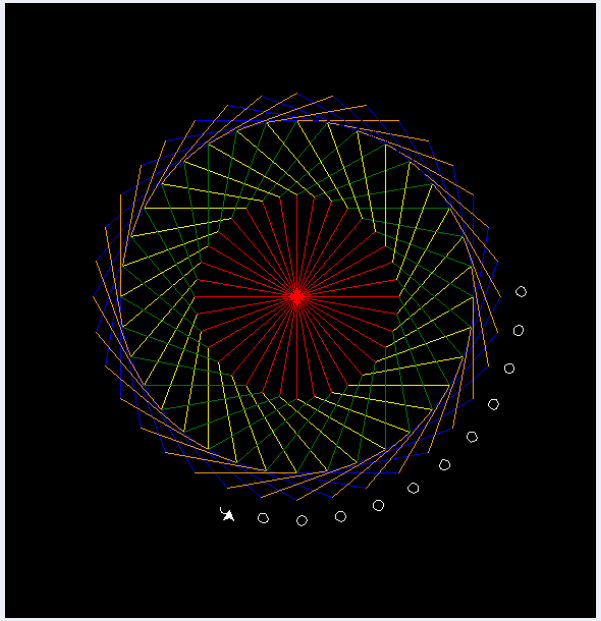
- The sample code below executes an artwork using cool turtle graphics. Turtle is also one of Python's standard libraries and is very easy to use.
- Descriptions for the Python Turtle Graphics Module is in the link below
- https://docs.python.org/3/library/turtle.html?highlight=turtle#module-turtle.

```
# make a geometric rainbow pattern
import turtle
colors = ['red', 'yellow', 'blue', 'orange', 'green', 'red']

aiden= turtle.Turtle()
turtle.bgcolor('black') # turn background black
# make 36 hexagons, each 10 degrees apart

for n in range(36):
# make hexagon by repeating 6 times
    for i in range(6):
        aiden.color(colors[i]) # pick color at position i
        aiden.forward(100)
        aiden.left(60)
    # add a turn before the next hexagon
```

Q1. Complete the creative artwork by looking at the document with your colleagues.

```
    aiden.right(10)

# get ready to draw 36 circles
aiden.penup()
aiden.color('white')
# repeat 36 times to match the 36 hexagons
for i in range(36):
    aiden.forward(220)
    aiden.pendown()
    aiden.circle(5)
    aiden.penup()
    aiden.backward(220)
    aiden.right(10)
# hide turtle to finish the drawing
aiden.hideturtle()
```

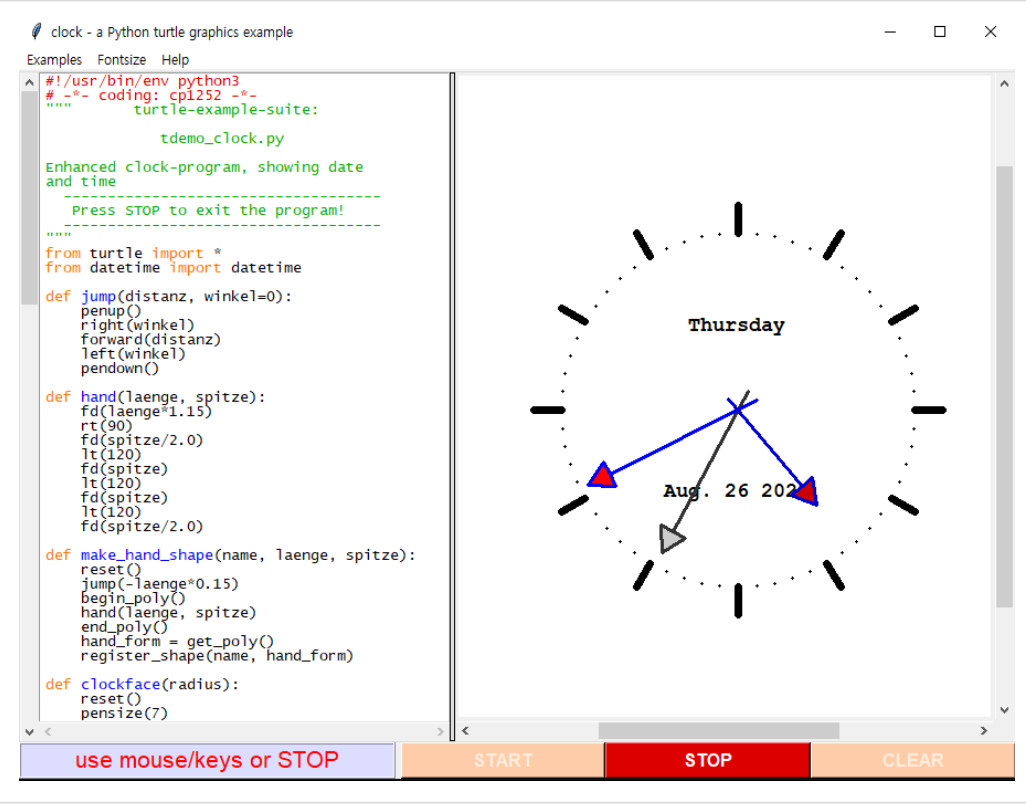
Q1. Complete the creative artwork by looking at the document with your colleagues.



Line 1, 6, 7, 10, 12, 15, 18, 21, 29

- 1: make a geometric rainbow pattern
- 6: turn background black
- 7: make 36 hexagons, each 10 degrees apart
- 10: make hexagon by repeating 6 times
- 12: pick color at position i
- 15: add a turn before the next hexagon
- 18: get ready to draw 36 circles
- 21: repeat 36 times to match the 36 hexagons
- 29: hide turtle to finish the drawing

Q1. Complete the creative artwork by looking at the document with your colleagues.



When you run the help → turtle menu in the menu selection in Python Idle, the demo window, as shown on the left, is executed. You can take a look at this one by one with your pair programming colleague and choose a demo you want to use to change the artwork.