Leena Kirtikar
SE-IT S13
Roll No. 43

Write python program to install Python (Jupyter Notebook), explain the various datatypes in Python and also demonstrate decision making and control statements.

I.)To install Python (Jupyter Notebook)

```
Anaconda conveniently installs Python, the Jupyter Notebook, and other commonly us
ed packages for scientific computing and data science.
```

Use the following installation steps:

1. Download Anaconda. Recommended: Anaconda’s latest Python 3 version (currently Python 3.5).
2. Accept the terms and conditions and install the version of Anaconda which you downloaded.
3. Now the Jupyter Notebook has been installed.

To run the notebook:

1. Open the command prompt.
2. Change the directory to where you want to save your python files.
3. Type "jupyter notebook" to launch the notebook.
4. Now the notebook has been launched and you can statrt coding.

II.)The various datatypes in Python are (standard or built-in data types):

- Numeric
- Sequence Type
- Boolean
- Set
- Dictionary

a.)Numeric: In Python, numeric data type represent the data which has numeric value. Numeric value can be integer, floating number or even complex numbers. These values are defined as int, float and complex class in Python.

1.)Integer: This value is represented by int class. It contains positive or negative whole numbers (without fraction or decimal). In Python there is no limit to how long an integer value can be.

In [3]:

```
x = 5
print("Type of x is: ", type(x))
```

```
Type of x is:  <class 'int'>
```

2.)Float: This value is represented by float class. It is a real number with floating point representation. It is specified by a decimal point. Optionally, the character e or E followed by a positive or negative integer may be appended to specify scientific notation.

In [4]:

```
y = 5.0
print("\nType of y is: ", type(y))
```

```

Type of y is:  <class 'float'>
```

3.)Complex Numbers: Complex number is represented by complex class. It is specified as (real part) + (imaginary part)j.

In [5]:

```
z = 5 + 10j
print("\nType of y is: ", type(y))
```

```

Type of y is:  <class 'float'>
```

b.)Sequence Type: In Python, sequence is the ordered collection of similar or different data types. Sequences allows to store multiple values in an organized and efficient fashion. There are several sequence types in Python:

1.)String: In Python, Strings are arrays of bytes representing Unicode characters. A string is a collection of one or more characters put in a single quote, double-quote or triple quote. In python there is no character data type, a character is a string of length one. It is represented by str class. Strings are immutable. It means you can't change an existing string.

In [7]:

```
s="Thadomal Shahani Engineering College"
print("\nType of s is: ", type(s))
```

```

Type of s is:  <class 'str'>
```

2.)List: List is an ordered sequence of items. Lists are just like the arrays, declared in other languages which is a ordered collection of data. It is one of the most used datatype in Python and is very flexible. All the items in a list do not need to be of the same type. Lists are mutable i.e. flexible. A list can be modified after it is created.

In [9]:

```
l=['apple',12,2.5,'peach',('hello','world'),['rain']]
print("\nType of l is: ", type(l))
```

```

Type of l is:  <class 'list'>
```

3.)Tuple: Just like list, tuple is also an ordered collection of Python objects. It can contain any type of variables. The only difference between tuple and list is that tuples are immutable i.e. not flexible. Tuple cannot be modified after it is created. Tuples are used to write protected data and are usually faster than lists as they cannot change dynamically. It is represented by tuple class.

In [10]:

```
t=('apple',12,2.5,'peach',('hello','world'),['rain'])
print("\nType of t is: ", type(t))
```

```

Type of t is:  <class 'tuple'>
```

c.)Boolean: Data type with one of the two built-in values, True or False. Boolean objects that are equal to True are truthy (true), and those equal to False are falsy (false). But non-Boolean objects can be evaluated in Boolean context as well and determined to be true or false. It is denoted by the class bool.

In [11]:

```
print(type(True))
print(type(False))
```

```
<class 'bool'>
<class 'bool'>
```

d.)Set: Set is an unordered collection of data type that is iterable, mutable and has no duplicate elements. The order of elements in a set is undefined though it may consist of various elements. Set is defined by values separated by comma inside braces { }. We can perform set operations like union, intersection on two sets. Sets have unique values. They eliminate duplicates. Since, set are unordered collection, indexing has no meaning. Hence, the slicing operator [] does not work.

In [12]:

```
a = {5,2,3,1,4}
print("\nType of a is: ", type(a))
```

```

Type of a is:  <class 'set'>
```

e.)Dictionary: Dictionary in Python is an unordered collection of data values, used to store data values like a map, which unlike other Data Types that hold only single value as an element, Dictionary holds key:value pair. It is generally used when we have a huge amount of data. Dictionaries are optimized for retrieving data. We must know the key to retrieve the value. Key-value is provided in the dictionary to make it more optimized. Each key-value pair in a Dictionary is separated by a colon : , whereas each key is separated by a 'comma'.

In [14]:

```
car = {  "brand": "Ford",  "model": "Mustang",  "year": 2021 }
print("\nType of car is: ", type(car))
```

```

Type of car is:  <class 'dict'>
```

III.)Write python programs to handle decision making and control statements.

A) Decision Making Statements:

1. if

Syntax:

```
if expression:
    statement(s)
```

if statement - If the boolean expression evaluates to TRUE, then the block of statement(s) inside the if statement is executed. If boolean expression evaluates to FALSE, then the first set of code after the end of the if statement(s) is executed.

2. if else

Syntax:

```
if expression:
    statement(s)
else:
    statement(s)
```

if else statement - An else statement can be combined with an if statement. An else statement contains the block of code that executes if the conditional expression in the if statement resolves to 0 or a FALSE value. The else statement is an optional statement and there could be at most only one else statement following if.

3. nested if else (elif)

Syntax:

nested if else - There may be a situation when you want to check for another condition after a condition resolves to true. In such a situation, you can use the nested if construct. In a nested if construct, you can have an if...elif...else construct inside another if...elif...else construct.

Q1.To check if a number is positive or negative .

In [25]:

```
#Demonstration of the if statement.

num = 25
if num > 0:
    print(num, "is a positive number.")

num = -1
if num < 0:
    print(num, "is a negative number.")
```

```
25 is a positive number.
-1 is a negative number.
```

Q2. To check if the number is even or odd.

In [23]:

```
#Demonstration of the if else statement.

number=int(input("Enter a number: "))
if number%2==0:
    print(number,"is an even number")
else:
    print(number,"is an odd number")
```

```
Enter a number: 44
44 is an even number
```

Q3. To check if the number is a whole number or a natural number.

In [13]:

```
#Demonstration of the elif statement.

number=int(input("Enter a number: "))
if number>=1:
    print(number,"is a natural number")
elif number>=0:
    print(number,"is a whole number")
```

```
Enter a number: 0
0 is a whole number
```

Q4.To find the greatest number between three numbers.

In [22]:

```
#Demonstration of nested if statements.

num1= input("Enter first number :")
num2= input("Enter Second number :")
num3= input("Enter third number :")
if(num1>num2):
    if(num1>num3):
        print(num1, "is the greatest")
    else:
        print(num3, "is the greatest")
elif(num2> num3):
    print(num2,"is the greatest")
else:
    print(num3, "is  the greatest")
```

```
Enter first number :5
Enter Second number :3
Enter third number :1
5 is the greatest
```

Q5. To calculate the grade of a student.

In [11]:

```
#Demonstration of the elif ladder.

avg=int(input("Enter a number: "))
if avg>=91 and avg<=100:
    print("Your Grade is A1")
elif avg>=81 and avg<91:
    print("Your Grade is A2")
elif avg>=71 and avg<81:
    print("Your Grade is B1")
elif avg>=61 and avg<71:
    print("Your Grade is B2")
elif avg>=51 and avg<61:
    print("Your Grade is C1")
elif avg>=41 and avg<51:
    print("Your Grade is C2")
elif avg>=33 and avg<41:
    print("Your Grade is D")
elif avg>=21 and avg<33:
    print("Your Grade is E1")
elif avg>=0 and avg<21:
    print("Your Grade is E2")
else:
    print("Invalid Input!")
```

```
Enter a number: 74
Your Grade is B1
```

B) Control Statements:

1. for loop

Syntax:

```
for iterating_var in sequence:
    statements(s)
```

for loop - A for loop is used for iterating over a sequence (that is either a list, a tuple, a dictionary, a set, or a string). This is less like the for keyword in other programming languages, and works more like an iterator method as found in other object-orientated programming languages. With the for loop we can execute a set of statements, once for each item in a list, tuple, set etc.

2. while loop :

Syntax:

```
while expression:
   statement(s)
```

while loop - A while loop statement in Python programming language repeatedly executes a target statement as long as a given condition is true. Here, statement(s) may be a single statement or a block of statements. The expressiom may be any condition, and true is any non-zero value. The loop iterates while the condition is true. When the condition becomes false, program control passes to the line immediately following the loop.

3. break statement :

Syntax:

#loop statements
break

break statement - The break statement in Python terminates the current loop and resumes execution at the next statement, just like the traditional break found in C. The most common use for break is when some external condition is triggered requiring a hasty exit from a loop. The break statement can be used in both while and for loops.

4. continue statement :

Syntax:

#loop statements
continue

continue statement - The continue statement in Python returns the control to the beginning of the while loop. The continue statement rejects all the remaining statements in the current iteration of the loop and moves the control back to the top of the loop. The continue statement can be used in both while and for loops.

Q1.) To make a left pyramid of natural numbers.

In [6]:

```
#Demonstration of the for loop.

a= int(input("Enter the number of rows in the Pyramid: "))
for i in range(0,a+1):
    for j in range(i):
        print("*",end="")
    print("\n")
```

```
Enter the number of rows in the Pyramid: 6


*

**

***

****

*****

******
```

Q2.) To print the multiplication table of a given number.

In [7]:

```
#Demonstration of the while loop.

a= int(input("Enter a number: "))
i=1
while i<=10:
    print(a,"x",i,"=",a*i)
    i+=1
```

```
Enter a number: 7
7 x 1 = 7
7 x 2 = 14
7 x 3 = 21
7 x 4 = 28
7 x 5 = 35
7 x 6 = 42
7 x 7 = 49
7 x 8 = 56
7 x 9 = 63
7 x 10 = 70
```

Q3.)To find a word in a list.

In [8]:

```
#Demonstration of the break statement.

l=["hello","welcome","congratulations","goodbye","sunshine"]
a=input("Enter a word you want to search: ")
for i in l:
    if a==i:
        print(a,"found in the list")
        break
```

```
Enter a word you want to search: goodbye
goodbye found in the list
```

Q4.)To calculate the sum of numbers (5 numbers max), If the user enters a negative number, it's not added to the result.

In [19]:

```
#Demonstration of the for continue statement.
sum1=0
for i in range(6):
    k=int(input("Enter a number: "))
    if k<0:
        continue
    sum1=sum1+k

print("The sum of the numbers is: ",sum1)
```

```
Enter a number: 4
Enter a number: 5
Enter a number: 6
Enter a number: -1
Enter a number: 2
Enter a number: 3
The sum of the numbers is:  20
```

In [ ]:

```

```