Python Basics

Objectives:

- Input and Output
- Basic Data Types
- More on Strings

1. Input & Output

1.1 Get User Input

You can ask user for some input using Python built-in `input()` function.

- When `input()` function is called, the program will stop and wait for user to key in some data.
- It is optional to add prompt as function parameter.
- It always returns a string value.

Exercise:

What is the output from `type(x)` command?

```
x = input('How old are you?  ')
x
type(x)
```

Learn more about the `input()` function from help:

```
help(input)
```

1.2 Print Out

Python provides a built-in function `print()` to display data to the console or a file object.

```
>>> help(print)
Help on built-in function print in module builtins:

print(...)
    print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)

    Prints the values to a stream, or to sys.stdout by default.
    Optional keyword arguments:
    file:  a file-like object (stream); defaults to the current sys.stdo
ut.
    sep:   string inserted between values, default a space.
    end:   string appended after the last value, default a newline.
    flush: whether to forcibly flush the stream.
```

Understand more about `print()` function:

- It can accept multiple values, and uses `sep` value to separate them in output. By default, the `sep` value is a space( ).
- It automatically appends an `end` string at the end. By default, the `end` value is a new line (`\n`).
- It outputs data to a file object defined in `file` , which has a default value `sys.stdout` (console).

Exercise:

```
print('hello', 'world')
print('have', 'a', 'good', 'day', sep='-')
```

2. Basic Data Types

In programming, a **data type** defines the type of a data and its expected behaviour.

- Variables are commonly associated with a particular data type.

2.1 Overview of Data Types

Python support many data types. Common data types are:

- Boolean
- Numbers
- String
- List
- Tuple
- Dictionary
- Set

Collection Data Types

List, Tuple, Dictionary and Set are 4 collection data types in Python.

- **List** is a collection which is ordered and changeable.
    - Allows duplicate members.
- **Tuple** is a collection which is ordered and unchangeable.
    - Allows duplicate members.
- **Dictionary** is a collection which is unordered, changeable and indexed.
    - No duplicate members.
- **Set** is a collection which is unordered and unindexed.
    - No duplicate members.

Function type() & isinstance()

The `type()` function can be used to check data type of a variable or a value.

The `isinstance()` function can be used to test the type of an object.

Exercise:

```
type(123)
type(1.23)
isinstance('abc', str)
```

2.2 Booleans

A Boolean variable can represent either `True` or `False`.

- Boolean values are commonly associated with comparison operators, which will be covered in another session.

Exercise:

```
x = (1==1)
y = (1=='1')
print(x, y)
type(x)
```

2.3 Numbers

Numbers in Python are classified into **int**, **float**, and **complex**.

- The **int** type can contain integer values
- The **float** type can contain decimal values
- The **complex** type can contain imaginery values. Add a “j” or “J” after a number to make it imaginary or complex.

Exercise:

```
type(123)
type(1.23)
type(1 + 2j)
```

Delimitor

For ease of reading, you can use `_` (underscore) to seperate a integer value.

- It doesn't affect actual value.

Exercise:

```
x = 123_456_789
x
```

Integer of Different Number Systems

We normally use Decimal (base 10) number system to reprsent an integer. Python also supports other number systems:

- Binary (base 2)
- Octal (base 8)
- Hexadecimal (base 16)

Exercise:

```
11
0b11
0o11
0x11
```

2.4 Strings

String is a sequence of one or more characters enclosed within either single quotes `' '` or double quotes `" "`.

Python also supports multi-line strings which require a triple quotation mark at the start and one at the end.

Exercise:

```
x = 'hello'
y = "world"
z = """hello
world
agian"""
print(x, y)
len(x)
print(z)
```

Escape Character `\`

Similiar to other programming langugae, Python uses `\` (backslash) as escape character in strings.

Here are some common escape characters that are represented using backslash notation.

Symbol	Character
`\"`	Double-quote (")
`\'`	Single-quote (')
`\n`	Line feed (LF) or new line
`\t`	Horizontal Tab (TAB)
`\\`	Backslash ()

Exercise:

```
'hello\tworld'
"hello\nworld\nagain"
'New line character is \\n'
'I\'m going home'
"I'd work hard"
'"Go home", he said'
```

Note: For last 2 examples, it uses combination of single-quote (`'`) and double-quote (`"`) instead of escape character.

2.5 List

A list is a collection which is ordered and changeable.

Exercise:

```
fruits = ["apple", "banana", "cucumber"]
fruits
fruits[1]
fruits[-1]
fruits[2] = 'cherry'
fruits.append("durian")
len(fruits)
fruits
```

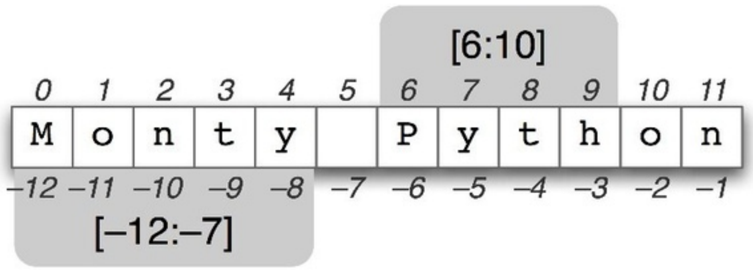
Create List

- Lists are writen with square brackets `[]` .

Access Items by Index

- List items can be accessed by its index number, which starts from 0.

- Negative indexing means beginning from the end, -1 refers to the last item, -2 refers to the second last item etc.



Change Item Value

- To change the value of a specific item, using the index number.

Exercise:

```
for x in fruits:
    print(x)
'banana' in fruits
'melon' in fruits
```

Loop Through a List

It is common to use `for-loop` to iterate through the list items.

Check if Item Exists

To determine if a specified item is present in a list, use the `in` keyword.

2.6 Tuple

A tuple is a collection which is ordered and **unchangeable**. Other than immutability, it behaves similarly to List.

- It is created using parenthesis `()`

Exercise:

```
fruits = ("apple", "banana", "cucumber")
fruits
for x in fruits:
    print(x)
'banana' in fruits
'melon' in fruits
```

2.7 Dictionary

A dictionary is a collection which is unordered, changeable and indexed.

- Each item in dictionary has a key and a value.
- It is created using curly brackets `{}`.
- Value in a dictionary can be accessed by its key.

Exercise:

```
d = {
  'date': '2020-01-02',
  "weekday": 'Thursday',
  "temp": 36
  }
d
d["date"]
```

3. Type Conversion

3.1 Dynamically and Strongly Typed

In a **statically typed** language, every variable name must be bound to a type when it is declared.

Python is a **dynamically typed** language. Type of a variable is only evaluated during run time.

- From all previous examples, all variables are created on the fly without declaring their data type.

In a **weakly typed language**, variables can be implicitly coerced to unrelated types. E.g. you can perform `"abc" + 99` in C#.

Python is a **strongly typed language**, variables of different type will not be automatically converted before operation.

Exercise:

```
'abc' + 99
'abc' + '99'
```

3.2 Type Casting

There may be times when you want to specify a type on to a variable. This can be done with casting.

Casting can be done using constructor functions:

- `int()` - constructs an integer number
- `float()` - constructs a float number
- `str()` - constructs a string

Exercise:

```
int(123)
int(1.23)
int('123')

float(123)
float("1.23")

str(123)
str(1.23)
str(123) + '456'
```

4. Recap

- What is the data type returned by input() function?
- How to print multiple values on console so that they are separated by space?
- How to convert a string to integer?
- How to check the data type of a variable?
- How to iterate through characters in a string?
- How to convert a variable from one data type to another data type?