Course	
Term	
Week	
Date	
Chapter. Topic	2.

Inputs and Outputs
print() and input() functions

Siva R Jasthi
Computer Science and Cybersecurity
Metropolitan State University

Outline

1. Taking Inputs input( ) function
2. Converting (casting) the string data
3. Printing Outputs print( ) function

variable vs function

The following are variables.

```
city_1
city_2
name_of_school
name_of_student
a
b
c
```

The following are functions.

```
print()
type()
input()
```

How can you tell whether something is a function or variable?

Is this a variable or a function? max

Is this a variable or a function? min()

Functions are usually actions = operations = verbs
Variables are usually nouns

input( ) function

input() **Function in Python**

Taking inputs from the user

We use **input**() function to take the inputs from the user.
Here are some examples.

```
name = input("Enter your name: ")
age = input(“Enter your age: “)
is_it_raining = input(“Is it raining?”);
```

Let us print the values and their types.

Anything you input is a STRING.

Taking inputs from the user

Write code in Python 3.6

```python
name = input("what is your name?")
age = input("what is your age?")
print(name)
print(age)

print("name type: ", type(name))
print("age type: ", type(age))


# I can not add an integer to a string
# I can not add a string to an integer
# However, I can add two strings
# and i can add two integers
a = 10 + 5
full_name = "Cuba" + "Jr."

age = int(age)
new_age = age + 10

print("You will be ", new_age, "in 10 years old")
```

Anything you input is a STRING.

Input() function and datatype

```
>>> name = input("Enter your name: ")
Enter your name: Balu
>>> print(name)
Balu
>>> type(name)
<class 'str'>
```

```
>>> age = input("Enter your age")
Enter your age50
>>> print(age)
50
>>> type(age)
<class 'str'>
>>>
```

```
>>> is_it_raining = input("Is it raining?")
Is it raining? False
>>> print(is_it_raining)
 False
>>> type(is_it_raining)
<class 'str'>
```

input() function and type

What is the conclusion?

input() function returns only strings.

By default, Python treats every input as STRING (str)

We need to do something to convert that to the proper data type.

```
>>> age = input("Enter your age")
Enter your age50
>>> print(age)
50
>>> type(age)
<class 'str'>
>>>
```

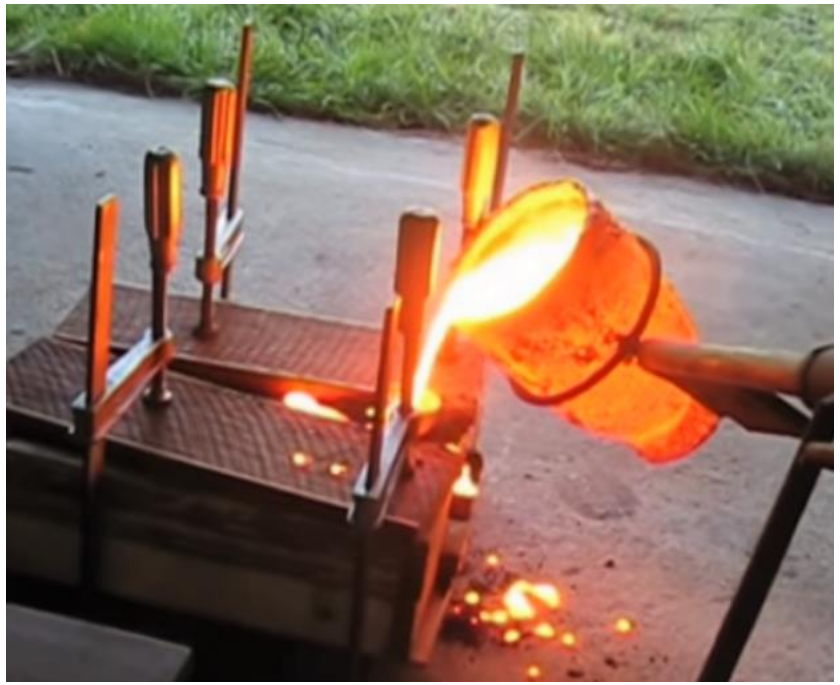
We need to cast the string input into integer
(casting = pouring = fitting)

casting (pouring, fitting)

Casting a string input into an integer

```
>>>
>>> age = input("Enter your age? ")
Enter your age? 13
>>> print(age)
13
>>> type(age)
<class 'str'>
>>> age = int(age)
>>> print(age)
13
>>> type(age)
<class 'int'>
>>>
```

Casting



casting is done through these functions

```
int( )   → to cast into integers

float( )   → to cast into floats

str( )   → to cast into strings

bool( ) → to cast into booleans
```

https://www.w3schools.com/python/python_casting.asp

What happens if you don’t convert?

Let us a couple of examples

Refer to the google colab notebook ch2_concepts_

print( ) function

Function Name

Parameter

```
print ('hello world')
```

Open Parenthesis

Close Parenthesis

print( ) function signature

```
print(*args,  sep = '  ', end = '\n')
```

*args: I accept 0 or 1 or N arguments

sep: If you give me a value, I will use it.
If you don’t give me a value, I will use space as the default

end: If you give me a value, I will use it.
If you don’t give me a value, I will use new line character as the default

print( ) function signature

print(*args, sep = ' ', end = '\n')

Examples:

```
#@title Exploring print statement (sep and end)
#print(*args, sep = ' ', end = '\n')

a, b, c = 1, 2, 3
print(a, b, c)
print(a, b, c, sep = '')
print('hi', 'how', 'are', 'you', sep='.')


print(100, 200, end = '*')
print(300)
```

print( ): It prints! Duh!

print( ) function is what we use to display the output

```
# a is the age of Bob
a = 13

# b is the age of John
b = 18

#printing the value of a
print(a)

#printing the value of b
print(b)
```

print ( ): It prints all types of variables

```
# school_name holds the name of the school
school_name = “Metro"

# student_name holds the name of a student
student_name = “John"

#printing the school name
print(school_name)

#printing the student’s name
print(student_name)
```

print ( ): It helps to provide additional input

Comments are visible only to you (the programmers). Those are not visible to the end-users (users or customers who use your program).

So, we can add some additional **arguments** to the print function.
All the **arguments** are separated by commas.

```
#printing the school name
print("School Name ", school_name)

#printing the student's name
print("Student Name ", student_name)
```

What is a string literal?

Anything you put in single or double quotes in python is a literal String.

```
Python prints it as-is
```

All about print( )

Function

Arguments (keyword arguments; positional arguments)

sep

end

default values

escape characters (new line character, tab character)

print( )

Can you print the following text?

```
ab
X
Y

P

Q
```

Brute-force method

Using escape character

print( )

Can you print the following text?

```
George O'Connor

The little girl said, "I love icecream".

She screamed, "I don’t like broccoli"
```

quotes

Single quotes

Double Quotes

Triple Quotes

print( )

Can you print the following text?

```
a, b, c, d = 1, 2, 3, 4
```

```
a, b, c, d = 1, 2, 3, 4
print(a, b, c, d)
print(a, b, c, d,sep='')
print(str(a) + str(b) + str(c) + str(d))
```

print( ): escape characters

New Line Character (/n)

Tab Character (/t)

Summary: What did we learn today?

- Arithmetic operators

print ( ): It prints all types of variables

```
# school_name holds the name of the school
school_name = “Metro"

# student_name holds the name of a student
student_name = “John"

#printing the school name
print(school_name)

#printing the student’s name
print(student_name)
```

print ( ): It helps to provide additional input

Comments are visible only to you (the programmers). Those are not visible to the end-users (users or customers who use your program).

So, we can add some additional **arguments** to the print function.
All the **arguments** are separated by commas.

```
#printing the school name
print("School Name ", school_name)

#printing the student's name
print("Student Name ", student_name)
```

What is a string literal?

Anything you put in single or double quotes in python is a literal String.

```
Python prints it as-is
```

All about print( )

Function

Arguments (keyword arguments; positional arguments)

sep

end

default values

escape characters (new line character, tab character)

print( )

Can you print the following text?

```
ab
X
Y

P

Q
```

Brute-force method

Using escape character

print( )

Can you print the following text?

```
George O'Connor

The little girl said, "I love icecream".

She screamed, "I don’t like broccoli"
```

print( )

Can you print the following text?

```
a, b, c, d = 1, 2, 3, 4
```

```
a, b, c, d = 1, 2, 3, 4
print(a, b, c, d)
print(a, b, c, d,sep='')
print(str(a) + str(b) + str(c) + str(d))
```

print(some_values, sep = ??, end = ??)

```
Using sep

Using end
```

What functions did we cover today?

type( )
print( )
input( )
int( )
float( )
bool( )
str( )

Built-in Functions

A
abs()
all()
any()
ascii()

B
bin()
bool()
breakpoint()
bytearray()
bytes()

C
callable()
chr()
classmethod()
compile()
complex()

D
delattr()
dict()
dir()
divmod()

E
enumerate()
eval()
exec()

F
filter()
float()
format()
frozenset()

G
getattr()
globals()

H
hasattr()
hash()
help()
hex()

I
id()
input()
int()
isinstance()
issubclass()
iter()

L
len()
list()
locals()

M
map()
max()
memoryview()
min()

N
next()

O
object()
oct()
open()
ord()

P
pow()
print()
property()

R
range()
repr()
reversed()
round()

S
set()
setattr()
slice()
sorted()
staticmethod()
str()
sum()
super()

T
tuple()
type()

V
vars()

Z
zip()

_
__import__()