Week 3: Tuesday

I-P-0

Input Process Output

Accepting numerics (type casting)

- `input()` always returns a string
- To accept a number, convert it to the desired type
- `int()` converts to integer
- `float()` converts to floating point

Example

```
age = input("Enter your age: ")
age = int(age)

# or

age = int(input("Enter your age: "))
```

Input Validation	Definition
`str.isalpha()`	True if all alphabetic
`str.isdigit()`	True if all digits
`str.isalnum()`	True if all alphanumeric
`str.isdecimal()`	True if all 0-9

Input Validation	Definition
`str.isnumeric()`	`True` if all numeric
`str.islower()`	`True` if all lowercase
`str.isupper()`	`True` if all uppercase

Example

```
# imagine these are inputs, This won't let me do an input
name = "something"
age = "20"


if name.isalpha():
    print("name is alpha")
else:
    print("name is not alpha")

if age.isdigit():
    print("age is digit")
else:
    print("age is not digit")

if age.isdecimal():
    print("age is decimal")
else:
    print("age is not decimal")
```

```
name is alpha
age is digit
age is decimal
```

Try Except validation, the best kind

- Since we are talking about validation, we can also use `try` and `except` to catch errors.
- `try except` tries to do something and something else if it fails.
- It handles errors gracefully without breaking everything.
- Different Errors: `ValueError`, `TypeError`, `IndexError`.
- See documentation for more errors.

Try Except, Example 1 (Good)

```python
try:
    age = "12.0"
    print(age.isnumeric())
    if not age.isnumeric():
        raise ValueError("Age is not numeric")
    print("Age is", age)
except ValueError as e:
    print(e)
```

```
False
Age is not numeric
```

Try Except, Example 2 (bad)

```python
try:
    age = "120"
    if not age.isnumeric():
        raise ValueError("Age is not numeric")
except ValueError as e:
    print(e)

print("Age is ", age)
```

```
Age is  120
```

Try Except, Example 3

```python
try:
    upper = "HELLo"
    if not upper.isupper():
        raise ValueError("Not all uppercase")
    print(f"{upper} is all uppercase")
except Exception as e:
    print(e)
```

```
Not all uppercase
```

Processing

- Doing stuff to the stuff
- it's the verb in the IPO: `input`, `process`, `output`
- input = information, process = action, output = result
- Could be simple:
  - Adding, concatinating, subtracting, multiplying, dividing, averaging, etc.
- or complex:
  - Data Cleaning, Data wrangling, transforming, modeling, etc.

Processing Example

```python
information = [i for i in range(1, 33, 3)]  # input
avg = sum(information) / len(information)  # process
print(f"information = {information}")  # output
print(f"Average of information is {avg}")  # output
```

```
information = [1, 4, 7, 10, 13, 16, 19, 22, 25, 28, 31]
Average of information is 16.0
```

f strings

- f strings are a way to format strings
- `f"string {variable} string"`

```python
name = "John"
age = 20
print(f"Hello, my name is {name} and I am {age} years old.")
```

```
Hello, my name is John and I am 20 years old.
```

String Functions

String Method	Explanation
`str.capitalize()`	“homer” → “Homer”
`str.upper()`	“homer” → “HOMER”
`str.lower()`	“HOMER” → “homer”
`str.title()`	“homer simpson” → “Homer Simpson”
`str.swapcase()`	“Homer Simpson” → “hOMER sIMPSON”
`str.strip()`	” homer ” → “homer”

Example

```
name = "    homer SIMPSON    "
nameb = "homer simpson"
print("capitalize", name.capitalize())  # capitalizes first character
print("capitalize", nameb.capitalize())
print("upper", name.upper())
print("lower", name.lower())
print("title", name.title())
print("swapcase", name.swapcase())
print("strip", name.strip())
```

```
capitalize     homer simpson
capitalize Homer simpson
upper     HOMER SIMPSON
lower     homer simpson
title     Homer Simpson
swapcase     HOMER simpson
strip homer SIMPSON
```

String Functions	Explanation
len(string)	“something” → 9
string.count(“s”)	“something” → 1
min(string)	“something” → “e”
max(string)	“something” → “t”
“s” in “something”	`True` if “s” in “something”
“s” not in “something”	`True` if “s” not in “something”
“some” * 3	“some” → “somesomesome”

String Functions	Explanation

String Functions	Explanation
"some"[3]	"some" → "e"
"some"[1:3]	"some" → "om"
"some"[0:3:2]	"some" → "se"

Example

```
first_name = "Nathan"
last_name = "Michalewicz"
print("Length of first name is", len(first_name))
print("Count of 'a' in first name is", first_name.count("a"))
print("Min of first name is", min(first_name))
print("Max of first name is", max(first_name))
print("Is 'a' in first name?", "a" in first_name)
print("Is 'a' not in first name?", "a" not in first_name)
print("first name * 3 is", first_name * 3)
print("first name[3] is", first_name[3])
print("first name[1:3] is", first_name[1:3])
print("first name[0:3:2] is", first_name[0:3:2])
```

```
Length of first name is 6
Count of 'a' in first name is 2
Min of first name is N
Max of first name is t
Is 'a' in first name? True
Is 'a' not in first name? False
first name * 3 is NathanNathanNathan
first name[3] is h
first name[1:3] is at
first name[0:3:2] is Nt
```

More String Function	Explanation
`string.join(iterable)`	” “.join([”a”, ”b”, ”c”]) →”a b c”
`string.partition("o")`	“something” → (“s”, “o”, “mething”) (returns a 3-tuple)
`string.rpartition("o")`	“something” → (“so”, “m”, “ing”) (returns a 3-tuple)
`string.split(" ")`	“a b c” → [“a”, “b”, “c”] (returns a list)

More String Function	Explanation
`string.split(sep, max)`	returns list splitting until max intervals
`string.rsplit(sep, max)`	returns list splitting until max intervals backwards
`string.splitlines()`	returns list split at lines

Example

```
lname = "Michalewicz"
fname = "Nathan"
s = "a b c d e f g"
lines = "a\nb\nc\nd\ne\nf\ng"  # \n is a newline character
print(fname.partition("a"))
print(fname.rpartition("a"))
print(s.split(" ", 3))
print(s.rsplit(" ", 3))
print(lines.splitlines())
```

```
('N', 'a', 'than')
('Nath', 'a', 'n')
['a', 'b', 'c', 'd e f g']
['a b c d', 'e', 'f', 'g']
['a', 'b', 'c', 'd', 'e', 'f', 'g']
```

Spacing Functions	Explanation
`string.ljust(10)`	“a” → “a”
`string.rjust(10)`	“a” → ” a”
`string.center(10)`	“a” → ” a ”
`string.zfill(10)`	“a” → “oooooooooa”
`string.center(10, "-")`	“a” → “—-a—–”

Spacing Functions	Explanation
`string.expandtabs(10)`	“a b” → “a b”
`string.strip()`	” a ” → “a”
`string.lstrip()`	” a ” → “a”
`string.rstrip()`	” a ” → ” a”

Example

```
s = "a"
print("ljust", s.ljust(10))
print("rjust", s.rjust(10))
print("center", s.center(10))
print("zfill", s.zfill(10))
print("center", s.center(10, "-"))
print("expandtabs", "a\tb".expandtabs(10))
print(".strip", "  a  ".strip())
print(".lstrip", "  a  ".lstrip())
print(".rstrip", "  a  ".rstrip())
```

```
ljust a
rjust          a
center     a
zfill 000000000a
center ----a-----
expandtabs a         b
.strip a
.lstrip a
.rstrip   a
```

Find/Replace Functions	Examples
`string.find("s")`	returns lowest index of s
`string.rfind("s")`	returns highest occurrence of s
`string.index("s")`	returns first occurrence of s
`string.rindex("s")`	returns highest index of s
`string.replace("s", "t")`	replaces s with t
`string.replace("s", "t", 3)`	replaces s with t a max of 3 times.

Example

```
s = "Nathan"
print("find: ", s.find("a"))
print("rfind: ", s.rfind("a"))
print("index: ", s.index("a"))
print("rindex: ", s.rindex("a"))
print("replace: ", s.replace("a", "o"))
print("replace: ", s.replace("a", "o", 1))
```

```
find:  1
rfind:  4
index:  1
rindex:  4
replace:  Nothon
replace:  Nothan
```

Number Functions

Basic Functions	Explanation
`abs(x)`	absolute value of x
`max(x, y, z)`	maximum of x, y, z
`min(x, y, z)`	minimum of x, y, z
`pow(x, y)`	x to the power of y

Basic Functions	Explanation
`round(x [,n])`	rounds x to the nearest integer and rounds to n decimals

Example

```
number = -4.12312
l = [1, 2, 3, 4, 5]
t = (1, 2, 3, 4, 5)
print("abs: ", abs(number))
print("max: ", max(l))
print("max: ", max(t))
print("min: ", min(l))
print("pow: ", pow(number, 3))
print("round: ", round(number, 2))
```

```
abs:  4.12312
max:  5
max:  5
min:  1
pow:  -70.09352873155534
round:  -4.12
```

Advanced Functions	Explanation
`math.copysign(x, y)`	returns x with the sign of y
`math.fabs(x)`	absolute value of x
`math.factorial(x)`	factorial () of x
`math.floor(x)`	returns the floor of x
`math.ceil(x)`	returns the ceiling of x
`math.fmod(x, y)`	returns the remainder of x/y

Example

```python
import math  # IMPORTANT!!!! import math

x = -4
y = 3
z = 4.712

print("copysign: ", math.copysign(x, y))
print("fabs: ", math.fabs(x))
print("factorial: ", math.factorial(5))
print("floor: ", math.floor(z))
print("ceil: ", math.ceil(z))
```

```
copysign:  4.0
fabs:  4.0
factorial:  120
floor:  4
ceil:  5
```

- We will not go over the `math` module in greater detail. We have classes for that.
- See the book for more detail.

---

random functions	Explanation
`random.choice(iter)`	random choice from iterable
`random.randint(x, y)`	random integer between x and y
`random.random()`	random float between 0 and 1
`random.randrange(x, y, z)`	random integer between x and y with step z
`random.shuffle(iter)`	shuffles iterable in place
`random.seed(x)`	sets the seed for random

Example

```python
import random  # IMPORTANT!!!! import random

l = [1, 2, 3, 4, 5]
r = []
print("choice: ", random.choice(l))
print("randint: ", random.randint(1, 10))
print("random: ", random.random())
print("randrange: ", random.randrange(1, 10, 2))
random.shuffle(l)
print("shuffle: ", l)
print("seed: ", r)
```

```
choice:  1
randint:  7
random:  0.4788783949238976
randrange:  1
shuffle:  [4, 3, 2, 5, 1]
seed:  []
```

Random Seed

- Random seed is a way to set the random number generator to a specific number, so it is reproducable

```python
random.seed(4)
print(random.randint(1, 100))
print(random.randint(1, 100))
```

```
31
39
```

```python
random.seed(4)
print(random.randint(1, 100))
print(random.randint(1, 100))
```

```
31
39
```

Datetime

- `datetime` is a module that allows you to work with dates and times
- `datetime.datetime.now()` returns the current date and time
- `datetime.datetime(year, month, day)` returns a date object
- `datetime.datetime.strptime(date, format)` returns a date object from a string
- typical format is `"%Y-%m-%d %H:%M:%S"`
- can also use `strftime` to format a date object

Why we need date objects

```
from datetime import datetime  # IMPORTANT!!!! import datetime

nathan = "1986-02-14"
rebecca = "1985-04-16"

# print(datetime.now())

if nathan > rebecca:
    print("nathan came later")
else:
    print("rebecca is older")
```

```
nathan came later
```

- But rebecca is older.

Why we need date objects, 2

- strptime() : string parse time
- strftime() : string format time

```
n = datetime.strptime(nathan, "%Y-%m-%d")
r = datetime.strptime(rebecca, "%Y-%m-%d")

if datetime.now() - n > datetime.now() - r:
    print(f"nathan is older, he was born {datetime.strftime(n, '%B %d, %Y')}")
else:
    print(f"rebecca is older, she was born {datetime.strftime(r, '%B %d, %Y')}")

print(f"In case you were wondering that was a {r.weekday()}")
```

```
rebecca is older, she was born April 16, 1985
In case you were wondering that was a 1
```

Weekdays return 0-6

- Monday is 0
- Sunday is 6

```
days = {
    0: "Monday",
    1: "Tuesday",
    2: "Wednesday",
    3: "Thursday",
    4: "Friday",
    5: "Saturday",
    6: "Sunday",
}

print(f"Rebecca was born on a {days[r.weekday()]} in {r.year}")
```

```
Rebecca was born on a Tuesday in 1985
```

---

str format	Explanation
`%a`	Weekday, short version
`%A`	Weekday, full version
`%w`	Weekday as a number 0-6
`%d`	Day of month 01-31
`%b`	Month name, short version
`%B`	Month name, full version
`%m`	Month as a number 01-12
`%y`	Year, short version, without century
`%Y`	Year, full version

---

str format	Explanation
`%H`	Hour 00-23
`%I`	Hour 00-12
`%p`	AM/PM
`%M`	Minute 00-59
`%S`	Second 00-59
`%f`	Microsecond 000000-999999
`%z`	UTC offset
`%Z`	Timezone

Example

```
print(datetime.strftime(r, "%A, %B %d, %Y"))
print(datetime.strftime(n, "%I:%M:%S %p"))
print(datetime.strftime(n, "%a, %Y, %B %d"))
```

```
Tuesday, April 16, 1985
12:00:00 AM
Fri, 1986, February 14
```

Time delta

- `timedelta` is a way to add or subtract time from a date object

```
from datetime import timedelta, datetime

if datetime.now() - (r + timedelta(days=365)) < datetime.now() - n:
    print(f"Only in this python world is Rebecca younger")
else:
    print("Rebecca is older")
```

```
Only in this python world is Rebecca younger
```