MEMORIAL UNIVERSITY

Faculty of Engineering and Applied Science

Department of Electrical and Computer Engineering
St. John's, NL Canada A1B 3X5
Tel: 709 864 8177 Fax: 709 864 4042
https://www.mun.ca/engineering/ece

Name:

Student ID:

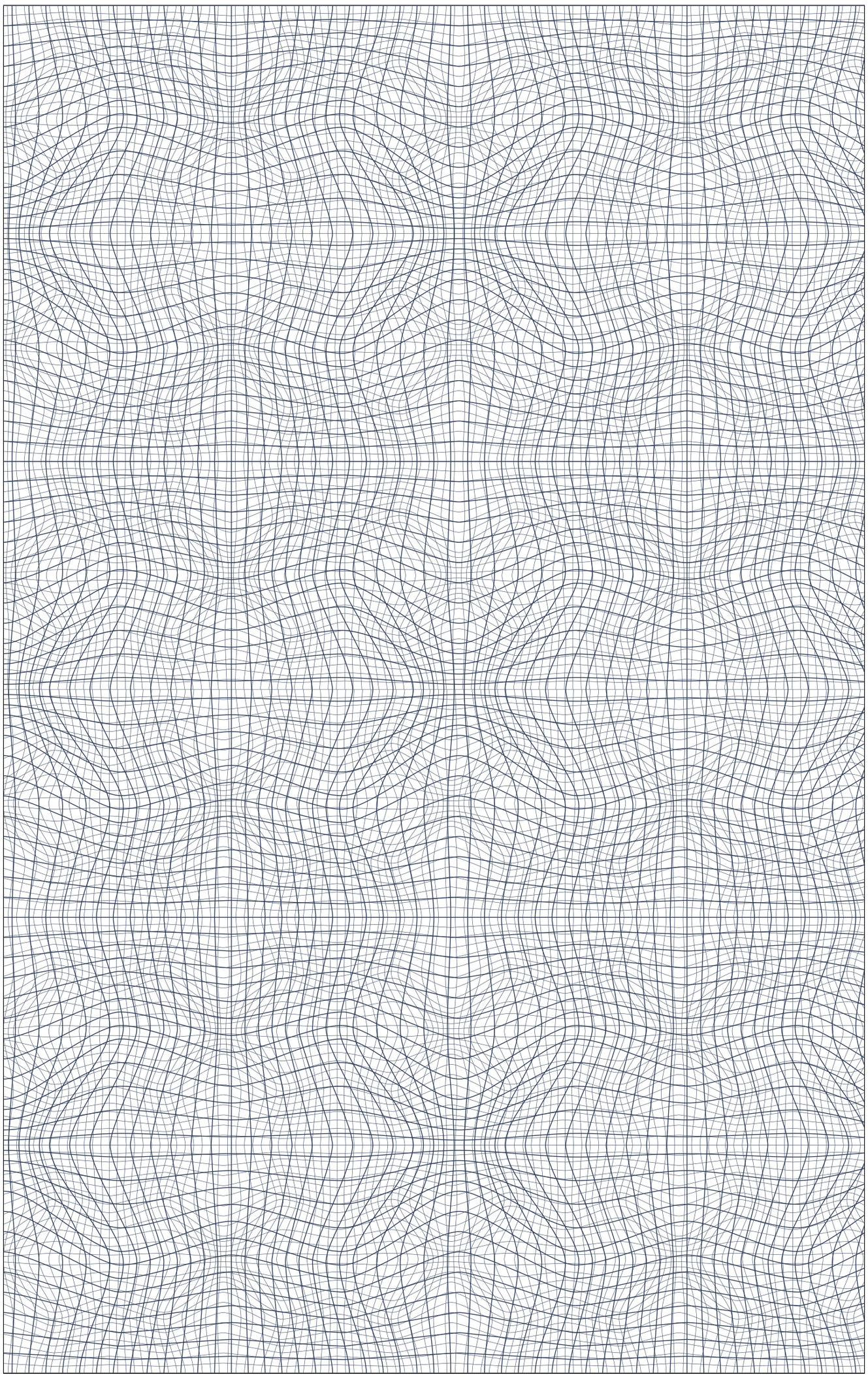
ENGI1020: Introduction to Programming

Final exam

Question	Points	Score
1	16	
2	20	
3	12	
4	12	
5	8	
6	28	
7	28	
Total:	124	

Instructions

1. Answer all questions.
2. Write your answers in the space provided on this paper.
3. Write your student number at the top of every answer page.
4. This is a closed-book exam: written aids are not permitted.
5. Calculators, phones and all other electronic aids are not permitted.
6. Unless otherwise specified, all code listed in this exam compiles and executes correctly.



Programming concepts

1. Choose the best answer for each of the following questions. **[16]**

(a) can change while a script runs [1]

◯ method ◯ syntax ◯ module ◯ function ◯ variable ◯ binary operator

(b) true or false [1]

◯ list ◯ Boolean ◯ dictionary ◯ tuple ◯ integer ◯ string

(c) looks like an identifier, but not allowed [1]

◯ identifier ◯ function ◯ string ◯ keyword ◯ parameter ◯ abstraction

(d) holds ones and zeroes [1]

◯ synthetic language ◯ expression ◯ operator ◯ keyword ◯ procedure ◯ memory

(e) how to use a function [1]

◯ instruction ◯ parameter ◯ call ◯ address ◯ increment ◯ floating-point

(f) `n += 1` [1]

◯ method ◯ object ◯ loop ◯ condition ◯ increment ◯ comparator

(g) `n < 2` [1]

◯ loop ◯ parameter ◯ method ◯ increment ◯ comparator ◯ local variable

(h) `n ** 3` [1]

◯ binary operator ◯ comparator ◯ method ◯ unary operator ◯ loop ◯ increment

(i) has methods [1]

◯ instruction ◯ module ◯ method ◯ function ◯ parameter ◯ object

(j) has statements [1]

◯ argument ◯ expression ◯ parameter ◯ function definition ◯ operator ◯ type

(k) assignment is an example [1]

◯ method ◯ unary ◯ function ◯ statement ◯ processor ◯ call

(l) change a variable's value [1]

◯ variable ◯ assignment ◯ semantics ◯ module ◯ call ◯ floating-point

(m) one thing for a CPU to do [1]

◯ argument ◯ instruction ◯ comment ◯ method ◯ object ◯ parameter

(n) avoid if possible [1]

◯ programming ◯ module ◯ global variable ◯ method ◯ object ◯ function

(o) **`list`**, **`set`**, **`tuple`** [1]

◯ local variable ◯ type ◯ method ◯ loop ◯ semantics ◯ statement

(p) where something is stored [1]

◯ statement ◯ keyword ◯ index ◯ I/O ◯ precedence ◯ increment

2. Operators and expressions **[20]**

(a) In the expression `4 * x ** 3 > y - 2 and z`, what order will the operations be carried out in? Please refer to the operations by name (multiplication, exponentiation, etc.). [5]

1. _____
2. _____
3. _____
4. _____
5. _____

(b) Given a variable `minutes` containing an integer number of minutes, write a Python **expression** that evaluates to a tuple containing hours and minutes. For example, if `minutes` is 75, your expression should evaluate to `(1,15)`. If `minutes` is 180, your expression should evaluate to `(3,0)`. [3]

Workings (optional):

Answer:

.....

(c) What makes a valid *identifier*? [2]

.....

.....

.....

.....

(d) What is the (decimal) value of the integer whose binary representation is 1101_2? [2]

_____

(e) What is the binary representation of the integer 213_{10}? [4]

Workings (optional):

Answer:

_____

(f) Given a floating-point number in which the *exponent* part is `0100` and the *fractional* part is `1100`, what is the value of the floating-point number $2^{e-3} \cdot f$? [4]

Workings (optional):

Answer:

_____

Student ID: _____

Analysis

3. Given the following function definition: [12]

```
def foo(x, b=2):
    i = 0
    n = b

    if x < 1:
        return 0

    while True:
        if x < n:
            return i + foo(x / n)

        i += 1
        n *= b
```

(a) What, at a high level, does the `foo` function do (e.g., compute a sine? count elements divisible by `b`)? 2

. . . . . .

(b) What is the meaning of `b=2` in this function definition? 2

. . . . . .

(c) Write four different tests for the `foo` function, writing four distinct calls to the `foo` function and the expected return value for each.

i. _____ ⇒ _____ 2

ii. _____ ⇒ _____ 2

iii. _____ ⇒ _____ 2

iv. _____ ⇒ _____ 2

4. Consider the Python module `foo.py`: **[12]**

```
words = ["Engineering", "1020", "is", "crazy", "fun"]
n = len(words)

def foo(word, course=1020):
    words = ["Engineering", str(course), "is", word.upper()]
    print(" ".join(words))

def bar(p=False):
    global n, words

    if p:
        words.reverse()
    else:
        words.remove("crazy")
        n = len(words)

    return words[-1]
```

(a) What will the following Python script output if it is run in the same directory as `foo.py`? [8]

```
from foo import *

print("!".join(words[::2]))

foo("challenging")
foo("static", 1010)

print([w for w in words if w.isnumeric()])
```

Workings (optional):

Answer:

.....

.....

.....

.....

(b) After the following script is executed from the same directory:

```
import foo

bar = foo.foo("superfun")
baz = foo.bar()
wibble = foo.bar(True)
```

Workings (optional):

Answer:

(c) The value of `bar` will be: _____ [1]

(d) The value of `baz` will be: _____ [1]

(e) The value of `wibble` will be: _____ [1]

(f) The value of `foo.n` will be: _____ [1]

Student ID: _____

5. **Syntactic and semantic errors** **[8]**

Identify four logical/semantic errors and four syntactic errors in the following Python module.

```
def sqrt(x=0, err):
    """Compute the square root of x to within a specified error."""

    while e < err:
        guess = 2.0

        div = x // guess
        e = guess - div
        if e < 0:
            e -= e

    return guess


# Some test code:
x = input("Enter a number: ")
y = sqrt(x)
if y * y != x:
    print(Error: y times y != x)
```

(a) Semantic error: 1

.....

(b) Semantic error: 1

.....

(c) Semantic error: 1

.....

(d) Semantic error: 1

.....

(e) Syntax error: 1

.....

(f) Syntax error: 1

.....

(g) Syntax error: 1

.....

(h) Syntax error: 1

.....

Program synthesis

6. **Itemized receipt** [28]

For this question, you must implement four Python functions that operate on a receipt of items. A receipt will be comprised of a **`list`** of items, each of which contains a description (a **`str`**, e.g., "bananas") and an amount. Each item may or may not be taxable.

(a) What type will you use to represent an item, and why? 2

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

(b) Define a function called `add_item`, which adds a single item to an existing **`list`**. `add_item` should accept up to four arguments: 2

- a **`list`** representating the receipt,
- a **`str`** representing the item description (e.g., "bananas"),
- a **`float`** representing the cost amount and
- a **`bool`** indicating whether or not the item is taxable (defaults to `True`).

For example, `add_item` may be called as follows:

```
import receipt

r = []
receipt.add_item(r, "Apples", 4.47, taxable=False)
receipt.add_item(r, "Bananas", 3.49, False)
receipt.add_item(r, "Cookies", 5.09)
receipt.add_item(r, "Bananas", 2.49, False)
```

Workings (optional):

Answer:

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

(c) Define a function called `tax`, which calculates the tax owed on a receipt's taxable items based on two arguments: 6

- a **`list`** representating the receipt and
- a **`float`** representing the tax rate.

`tax` should return the taxable amount **to the nearest cent** as a **`float`**. For example, the following script should output `0.51`:

```
import receipt

r = []
receipt.add_item(r, "Apples", 4.47, taxable=False)
receipt.add_item(r, "Bananas", 3.49, False)
receipt.add_item(r, "Cookies", 5.09)
print(receipt.tax(r, 0.1))
```

Workings (optional):

Answer:

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

(d) Define a function called `total` that calculates the total cost of the receipt (including taxes) based on up to two arguments: 8

- a **`list`** representating the receipt and
- a **`float`** representing the tax rate (defaults to `0.15`).

`total` should return **a tuple of dollars and cents** (both of which should be integers). For example, running the following Python script:

```
import receipt

r = []
receipt.add_item(r, "Apples", 4.47, taxable=False)
receipt.add_item(r, "Bananas", 3.49, False)
receipt.add_item(r, "Cookies", 5.09)
receipt.add_item(r, "Bananas", 2.49, False)
dollars, cents = receipt.total(r)

print(dollars, "dollars and", cents, "cents")
```

should produce the output `16 dollars and 30 cents .`

Workings (optional):

Answer:

Student ID: _____

(e) Define a function called `print_all` that prints a receipt, including the description and cost of each item as well as a capital "T" for taxable items. Finally, it should print the subtotal, tax and total. [10]

For bonus marks, the `print_all` function may also incorporate additional spaces to right-align the prices, as well as hyphen-based separator lines between the items, the subtotal, the tax and the total.

Basic output (full marks):

```
Tuition 11460.00
Fees 854.00
Insurance 690.12 T
Accommodations 7773.00 T
Meal plan 8115.00 T
Other 3600.00 T
Subtotal 21600.00
Tax 3026.72
Total 35518.84
```

"Fancy" output (bonus marks):

```
Tuition                 11,460.00
Fees                       854.00
Insurance                  690.12 T
Accommodations           7,773.00 T
Meal plan                8,115.00 T
Other                    3,600.00 T
--------------------------------
Subtotal                21,600.00
Tax                      3,026.72
--------------------------------
Total                   35,518.84
```

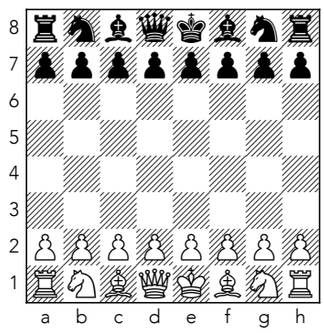
Workings (optional):

Answer:

7. **Computer chess** [28]

You are working as part of a team to create an application that plays chess against a human player. In this question, you must describe some **small** parts of this system. **You are not responsible for writing a complete chess program.**

(a) **User input** 8



Chess squares are referred to by *rank* (row) and *file* (column), as shown in the board to the right. Chess players describe squares on the board using a combination of the file and rank, e.g., `a1` for the bottom-left square or `h8` for the top-right square.

Write a Python function that will prompt the user for a square such as `f7` and convert that square into an integer rank and an integer file, both between 0 and 7 (inclusive). Your function should return the file and rank as a tuple. For example, if the user enters `a1`, your function should return `(0,0)`. If the user enters `d7`, your function should return `(3,6)`. If the user enters an invalid square, they should be prompted again until they enter a valid square.

You may write more than one function if it is helpful.

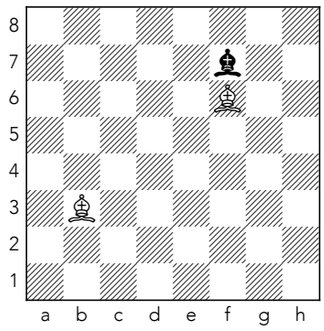
Workings (optional):

Answer:

Student ID: _____

(b) **Diagonal checking** 5

A chess *diagonal* is a line that covers the same number of ranks and files (i.e., it is a line with slope 1 or -1). For example, on the board depicted to the right, the bishops at b3 and f7 are on a diagonal from each other; those at b3 and f6 are not.



Using valid Python, implement a function that checks whether two squares (f_0, r_0) and (f_1, r_1) are on a diagonal together. You may assume (i.e., you do not need to check) that f_0, r_0, f_1 and r_1 are all integers between 0 and 7 (inclusive).

Examples:

- $(1, 2)$ to $(5, 6)$ **is** a diagonal
- $(1, 2)$ to $(5, 5)$ **is not** a diagonal
- $(6, 3)$ to $(4, 5)$ **is** a diagonal
- $(6, 3)$ to $(2, 5)$ **is not** a diagonal

Hint: you may find the built-in Python function `abs(x)` (absolute value) helpful.

Workings (optional):

Answer:

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

(c) Movement

15

A chess program must be able to distinguish legal from illegal moves. Each chess piece has a different set of legal moves available to it:

♔**The King (K)** can move one space at a time in any direction (including diagonally).

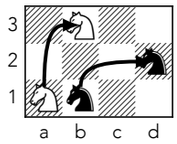
♕**The Queen (Q)** can move any number of spaces in a straight line in any direction (including diagonally).

♖**Rooks (R)** can move any number of spaces along a *rank* (horizonal row) or *file* (vertical row).

♗**Bishops (B)** can move any number of spaces along a diagonal.

♘**Knights (N)** move in an L shape:

1. two spaces along one rank or file,
2. a $90°$ turn, then
3. one more space along a different rank or file.



♙**Pawns (P)** can move one space **within a file** (note: this is a simplified version of the actual rules for a pawn, which are the most complicated)

Write a Python function that checks whether it is legal to move a piece of kind `piece` (`"K"`, `"Q"`, etc.) from location (f_0, r_0) to (f_1, r_1), where f_0, r_0, f_1 and r_1 are all integers between 0 and 7 (inclusive).

Your algorithm should be able to handle **any three of the six pieces described above**: you do not need to describe all six of them. You may use any functions that you have already defined in this question, and you may define any new abstractions that you find helpful.

Workings (optional):

Answer:

A Reference material

A.1 Common Arduino functions

```
def analog_read(pin):
    """Read a value from an analog port.

    Parameters:
      pin (int):     Analog pin to read from (e.g., 2 for A2).

    Returns:         Quantized analog value -(01023).
    """
```

```
def analog_write(pin, duty):
    """Write a value to an analog port.

    Parameters:
      pin (int):     Analog pin to write to (e.g., 2 for A2).
      duty (int):    Duty cycle to set -(0255).
    """
```

```
def digital_read(pin):
    """Read a value from a digital port.

    Parameters:
      pin (int):     Digital pin to read from (e.g., 4 for D4).

    Returns:         True or False.
    """
```

```
def digital_write(pin, value):
    """Write a value to a digital port.

    Parameters:
      pin (int):     Digital pin to write to (e.g., 4 for D4).
      value (bool):  Value to write (True or False).
    """
```

A.2 Python standard library

A.2.1 Select `math` functions and values

- `ceil(x)`
- `cos(x)`
- `degrees(x)`
- `e`
- `factorial(n)`
- `floor(x)`
- `inf`
- `isnan(x)`
- `log(x,base)`
- `nan`
- `pi`
- **`pow`**`(x)`
- `radians(x)`
- `sin(x)`
- `sqrt(x)`

A.2.2 `time.sleep(secs)`

Suspend execution of the calling thread for the given number of seconds. The argument may be a floating point number to indicate a more precise sleep time. The actual suspension time may be less than that requested because any caught signal will terminate the sleep() following execution of that signal's catching routine. Also, the suspension time may be longer than requested by an arbitrary amount because of the scheduling of other activity in the system.

A.2.3 Selected `list` methods

- `append`
- `clear`
- `count`
- `extend`
- `index`
- `insert`
- `remove`
- `reverse`
- `sort`

A.2.4 Selected `str` methods

- `capitalize`
- `count`
- `endswith`
- `find`
- `index`
- `isalnum`
- `isalpha`
- `isascii`
- `isdecimal`
- `isdigit`
- `isidentifier`
- `islower`
- `isnumeric`
- `isprintable`
- `isspace`
- `istitle`
- `isupper`
- `join`
- `lower`
- `lstrip`
- `replace`
- `rfind`
- `split`