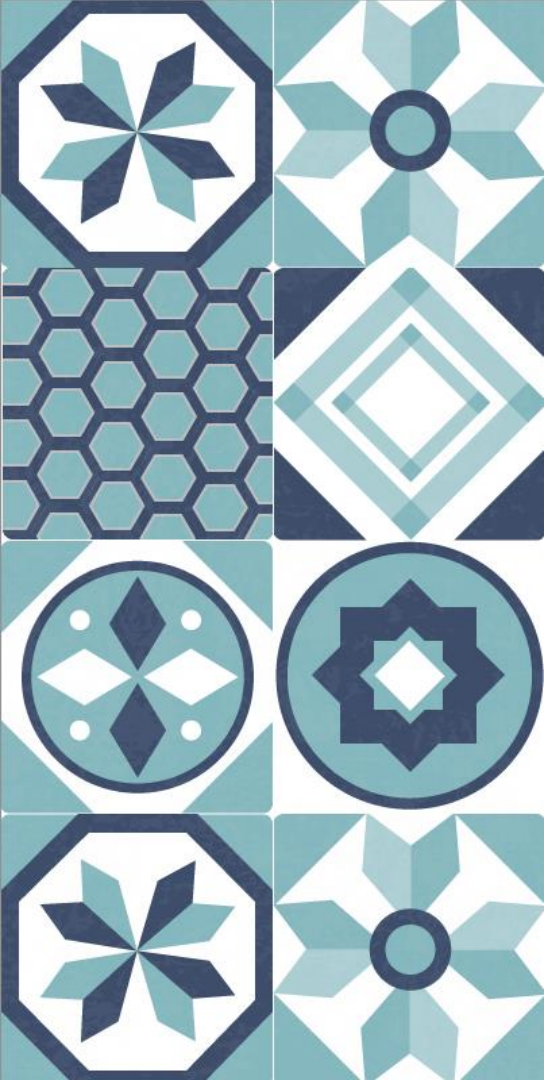


FONDAMENTI DI INFORMATICA

Alma Artis
Francesca Pratesi (ISTI, CNR)

CHI SONO

- Informatica laureate all'Università di Pisa nel 2013
- Dottorato in Informatica nel 2017
- Dal 2021 ricercatrice all'ISTI – CNR di Pisa
- Collaboratrice con l'Università di Pisa
  - Dipartimento di Informatica Umanistica
    - Fondamenti Teorici e Programmazione (co-docenza dal 2018; in precedenza tutor)
    - OFA-MAT (titolare nel 2018-2020 e 2021-2023)
  - Dipartimento di Fisica
    - Laboratorio di Informatica (co-docenza nel 2020-2022)
  - Dipartimento di Informatica (Master in Big Data & Social Mining)
    - Basi di dati (tutor dal 2016)
- Interessi di ricerca
  - Privacy e etica nell'Intelligenza Artificiale

CONTATTI

- Ricevimento: (da stabilire) lunedì pomeriggio dalle 15 alle 17 presso il CNR (meglio prendere appunamento via mail)
- Email: francesca.pratesi@isti.cnr.it (mettete il tag [AlmaArtis] nel soggetto della mail)
- Sito web del corso: https://github.com/prafra/AlmaArtis2022

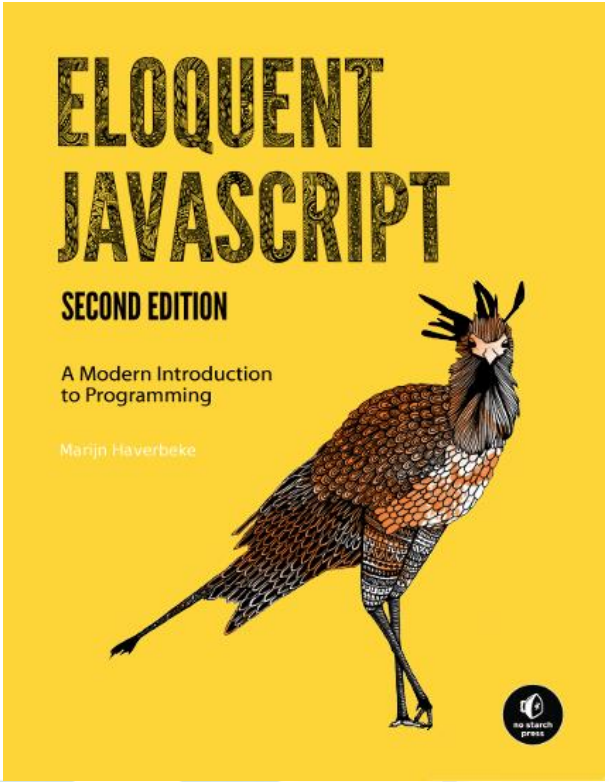
OBIETTIVI

- Concetti di base dell'Informatica
- Acquisizione di principi e strumenti di base della programmazione
- Elaborare la soluzione ad un problema attraverso la creazione di un algoritmo e la relativa codifica in un linguaggio di programmazione
- Il linguaggio di programmazione del corso sarà Java/Processing

PROGRAMMA DEL CORSO

- Parte 1: introduzione ai concetti di base dell'informatica
- Parte 2: elementi di programmazione con Java/Processing
- Parte 3: elementi di computer graphics e algoritmi avanzati
- Esercitazioni

LIBRI E RIFERIMENTI

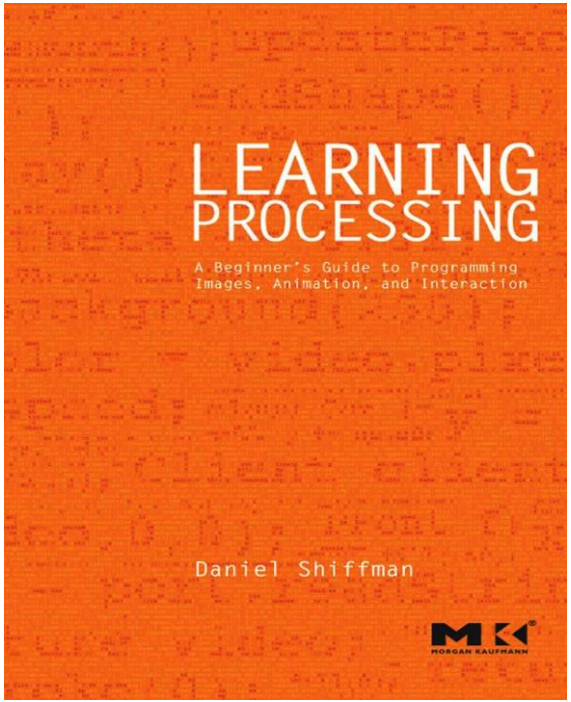


- Capitoli 1-4

Eloquent Javascript – Second Edition
Marijn Haverbeke

Available here: http://eloquentjavascript.net/

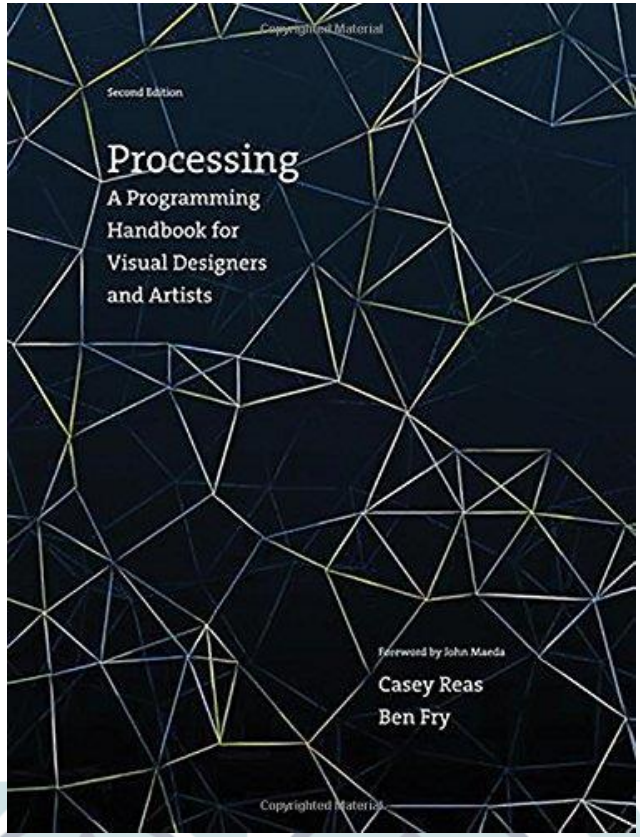
LIBRI E RIFERIMENTI



- Introduzione
- Capitolo 1-3

Learning Processing– Second Edition
Daniel Shiffman
Available here: http://learningprocessing.com/

LIBRI E RIFERIMENTI



Processing: Handbook for Visual Designers and Artists

Casey Reas, Ben Fry

ORGANIZZAZIONE DELLE LEZIONI

I Parte

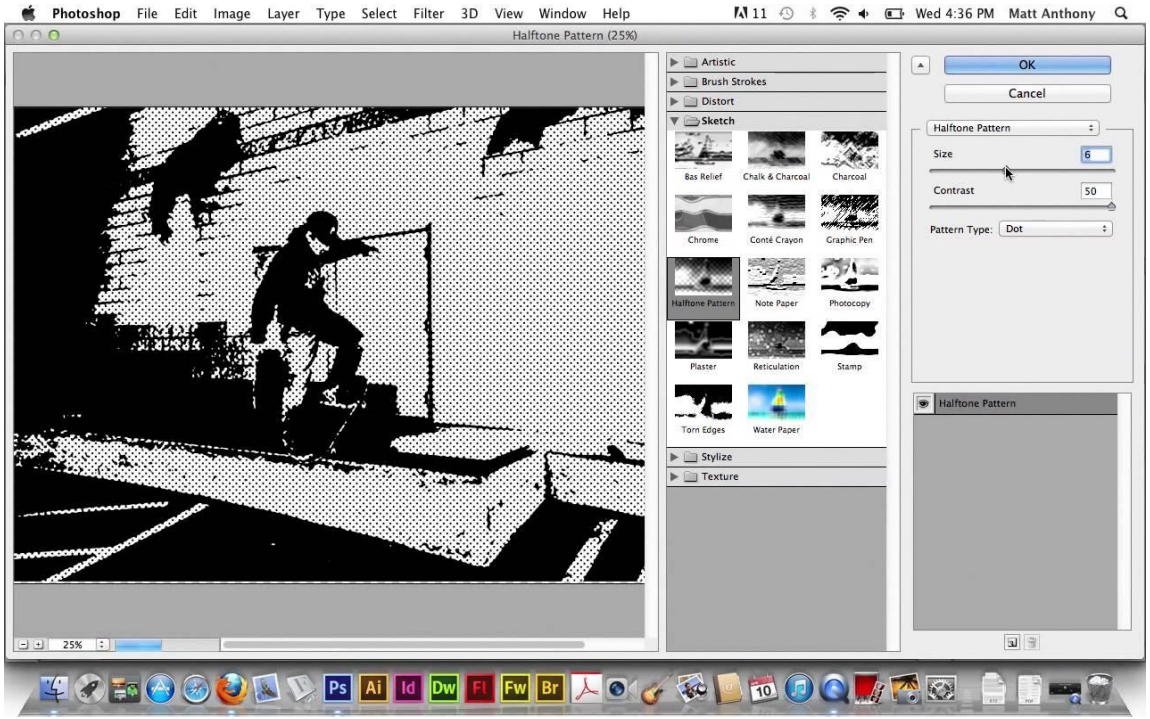
- Teoria
- Formazione
- Casi di studio

II Parte

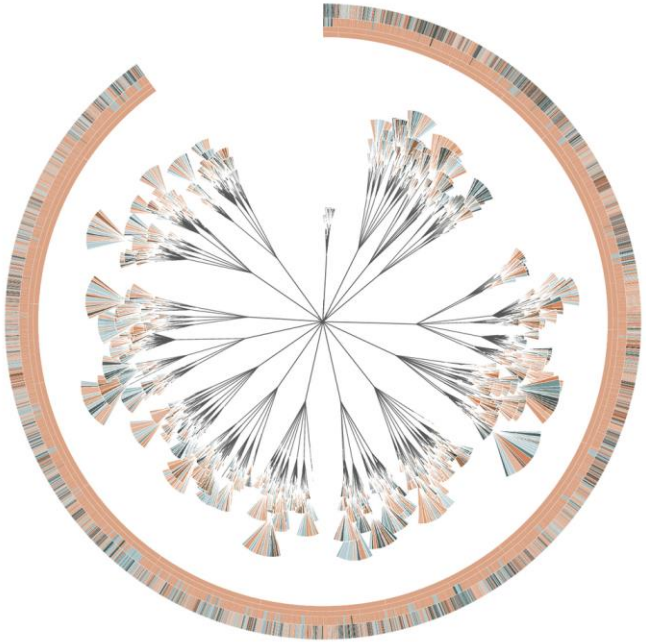
- Laboratorio
- Esercitazioni
- Esame scritto

PERCHÉ QUESTO CORSO?

PHOTOSHOP

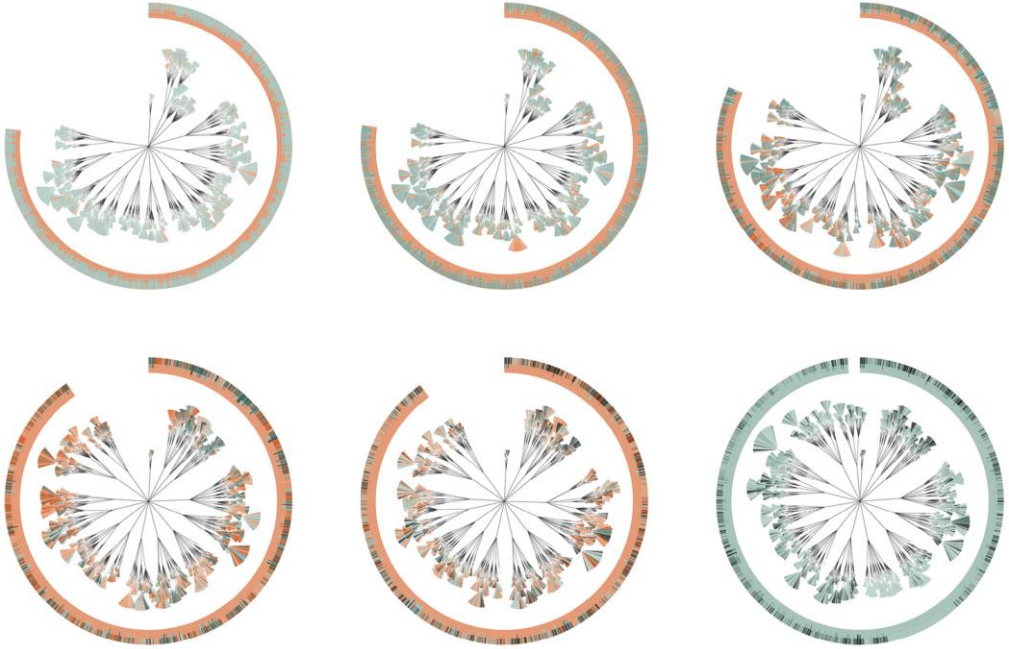

Photoshop File Edit Image Layer Type Select Filter 3D View Window Help
Wed 4:36 PM Matt Anthony
Halftone Pattern (25%)
Artistic
Brush Strokes
Distort
Sketch
Bas Relief
Chalk & Charcoal
Charcoal
Chrome
Conté Crayon
Graphic Pen
Halftone Pattern
Note Paper
Photocopy
Plaster
Reticulation
Stamp
Torn Edges
Water Paper
Stylize
Texture
OK
Cancel
Halftone Pattern
Size
6
Contrast
50
Pattern Type:
Dot
Halftone Pattern
25%

(EN)TANGLED WORD BANK



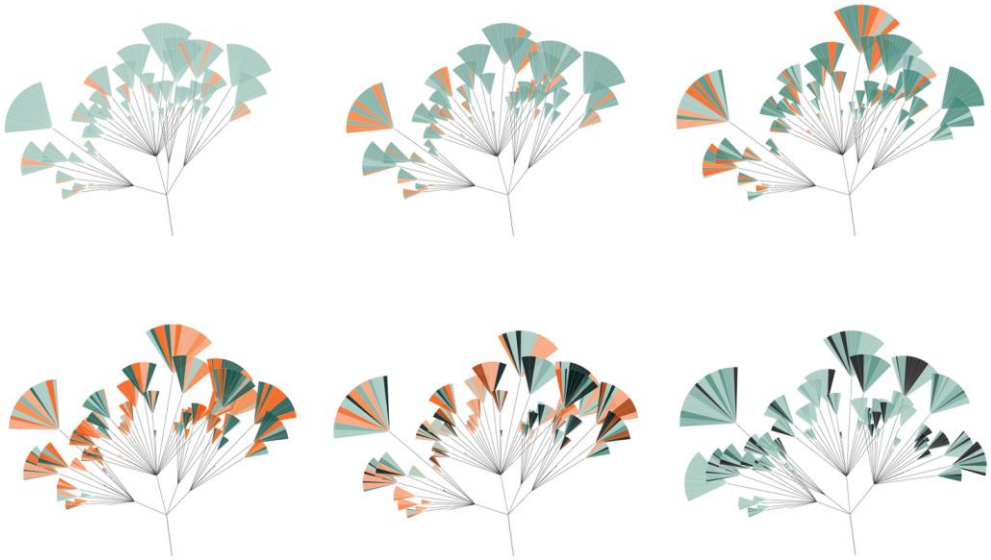
http://www.stefanieposavec.co.uk/entangled-word-bank/

(EN)TANGLED WORD BANK



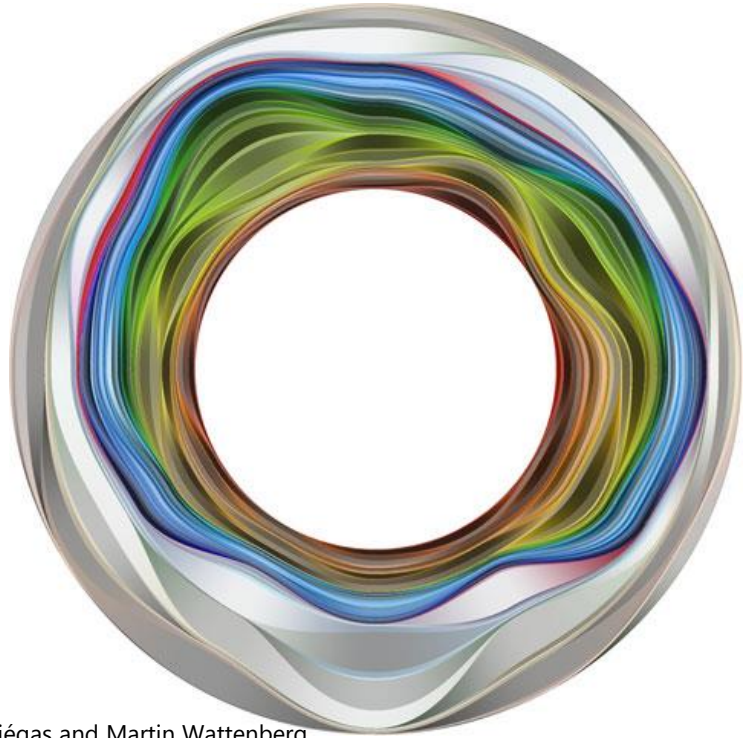
http://www.stefanieposavec.co.uk/entangled-word-bank/

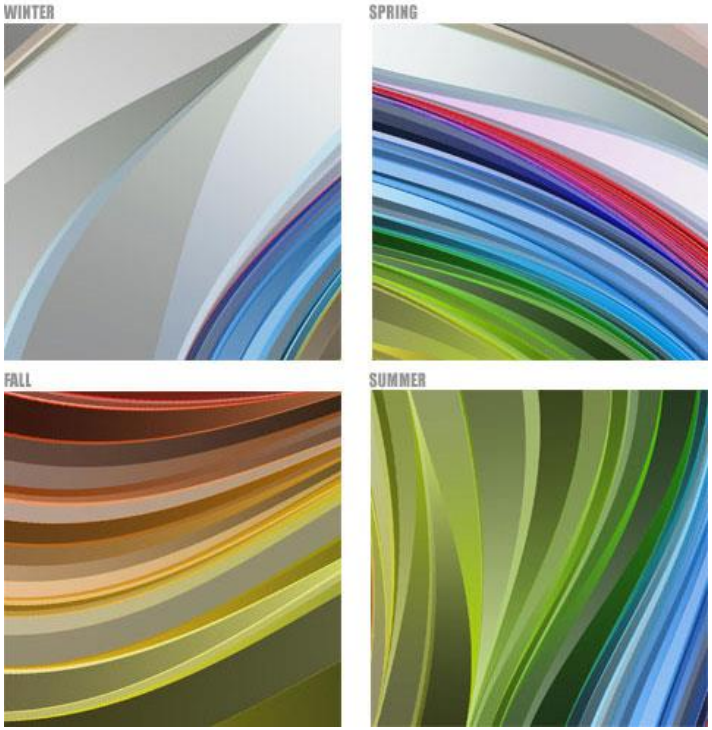
(EN)TANGLED WORD BANK



http://www.stefanieposavec.co.uk/entangled-word-bank/

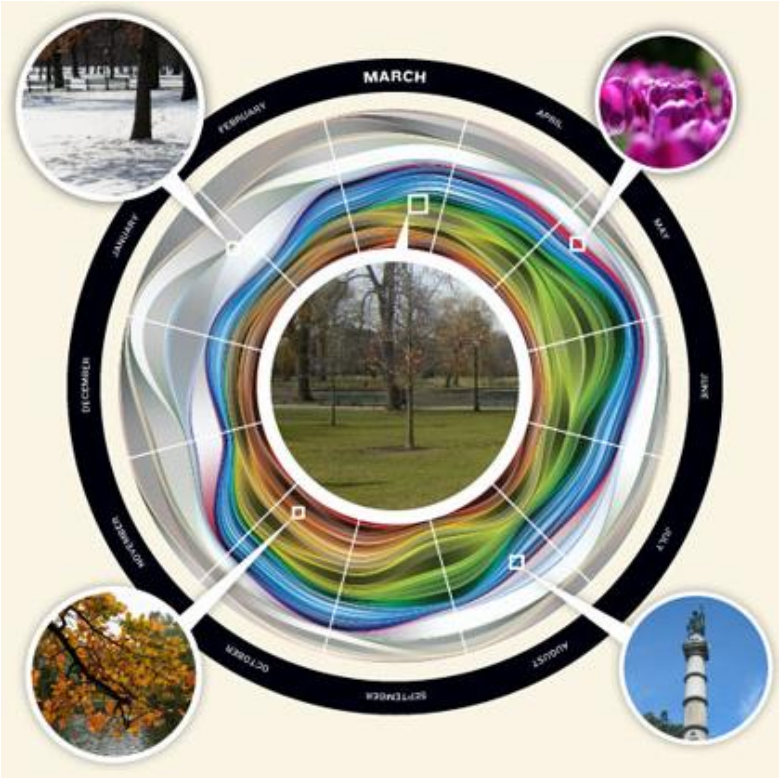
FLICKR FLOW





Fernanda Viégas and Martin Wattenberg.
http://hint.fm/about/

FLICKR FLOW



Fernanda Viégas and Martin Wattenberg.
http://hint.fm/projects/flickr/

WEB SEER

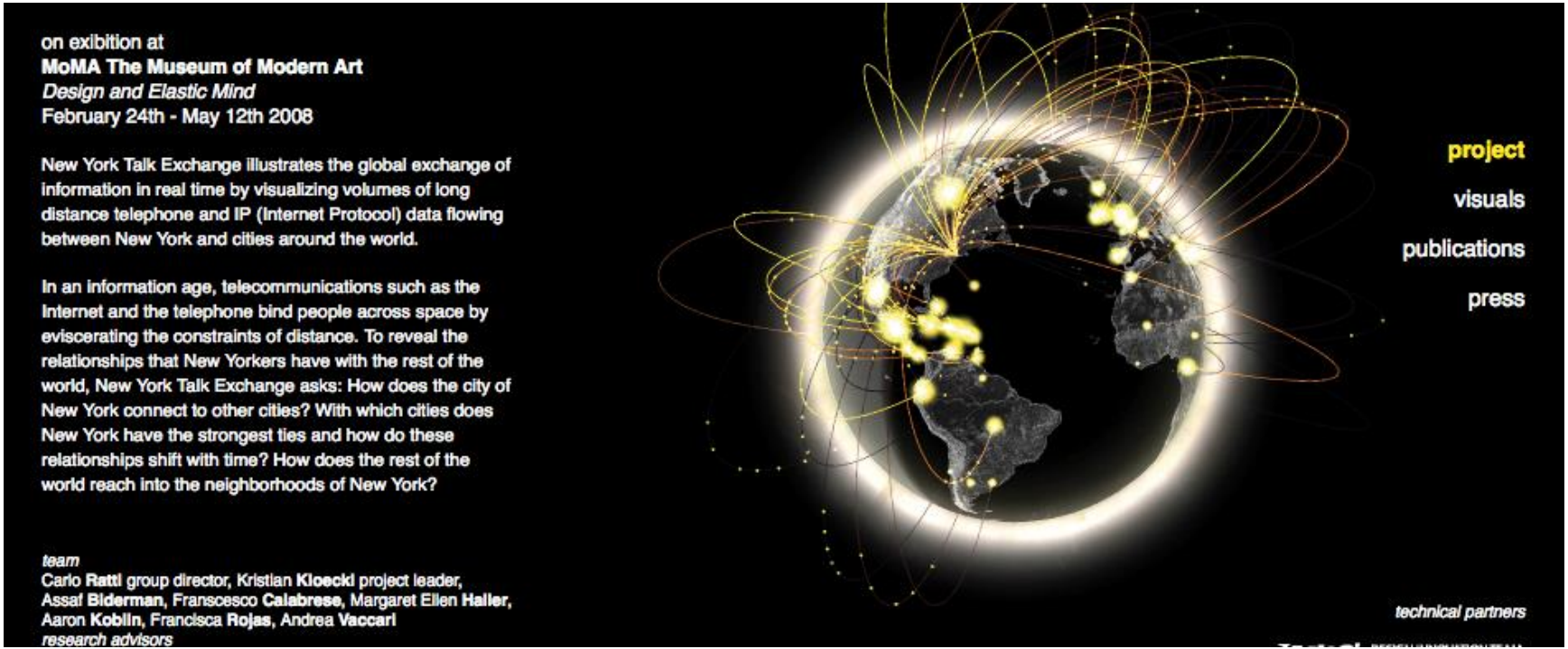
Fernanda Viégas and Martin Wattenberg.
http://hint.fm/projects/seer/

SILVERSTONE MOTOGP – OCTO EXHIBITION

CHRONOGRAPHS

http://reas.com/chronograph/

NYTE



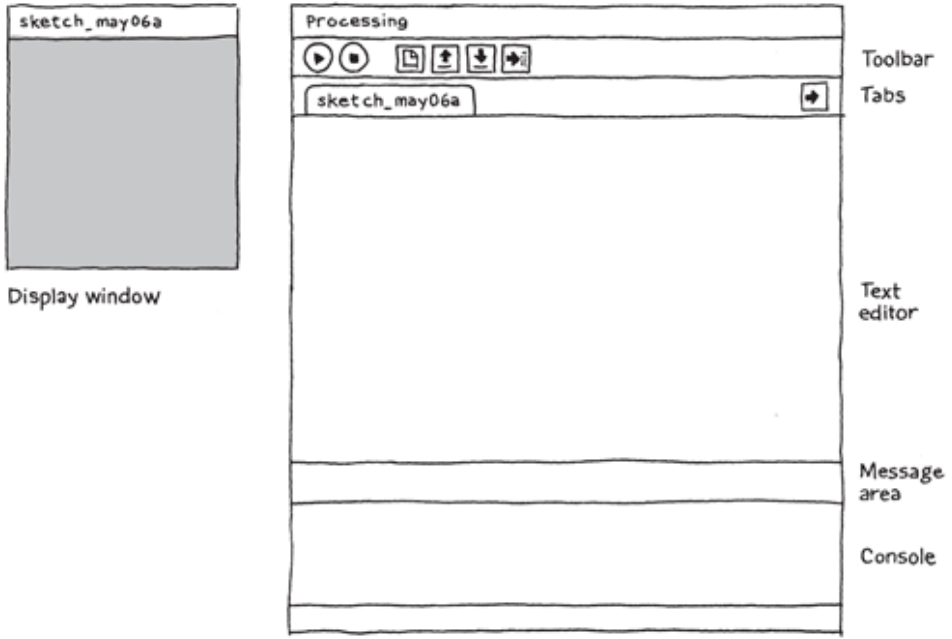
http://senseable.mit.edu/nyte/

INTRODUCING PROCESSING

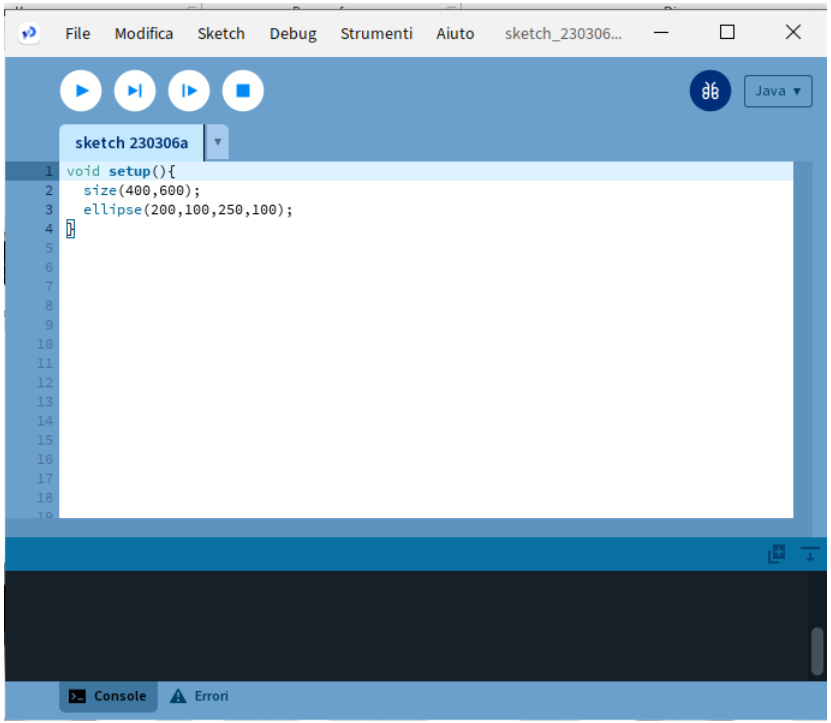
PROCESSING.ORG - INSTALLAZIONE

- Scaricare l'IDE da http://processing.org
- Estrarre tutti i file
- Eseguire l'aplicazione processing.exe

ANATOMY OF CODE EDITOR

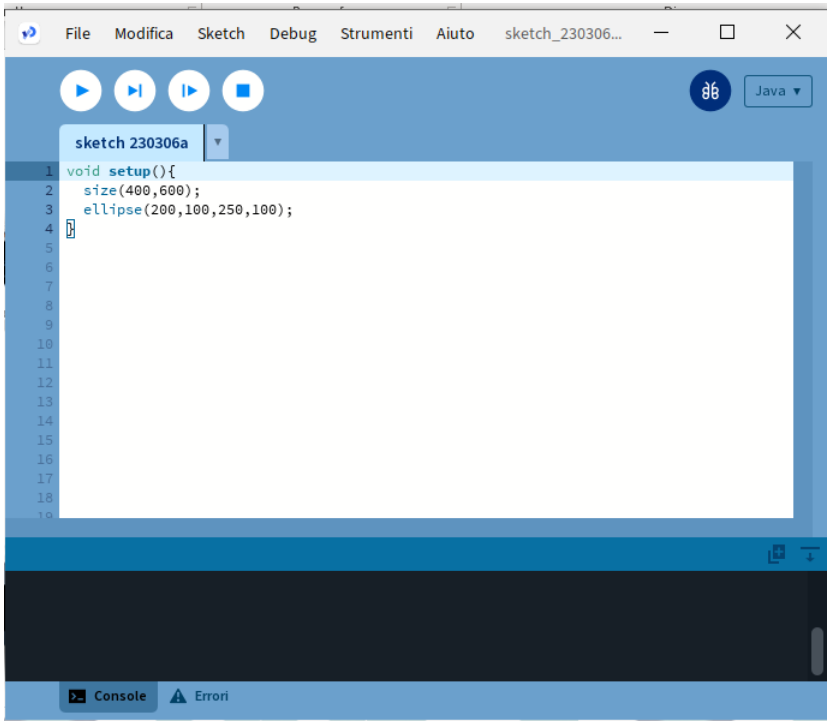


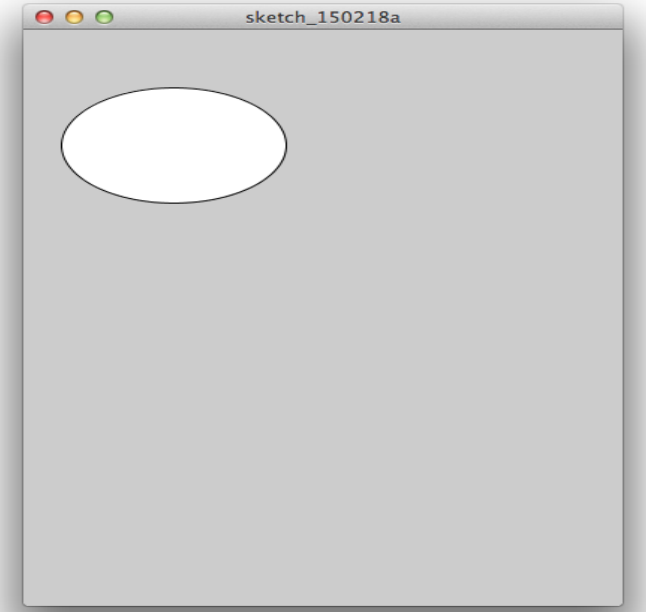
IL NOSTRO PRIMO SKETCH




```
void setup(){
    size(400,600);
    ellipse(200,100,250,100);
}
```

IL NOSTRO PRIMO SKETCH (2)





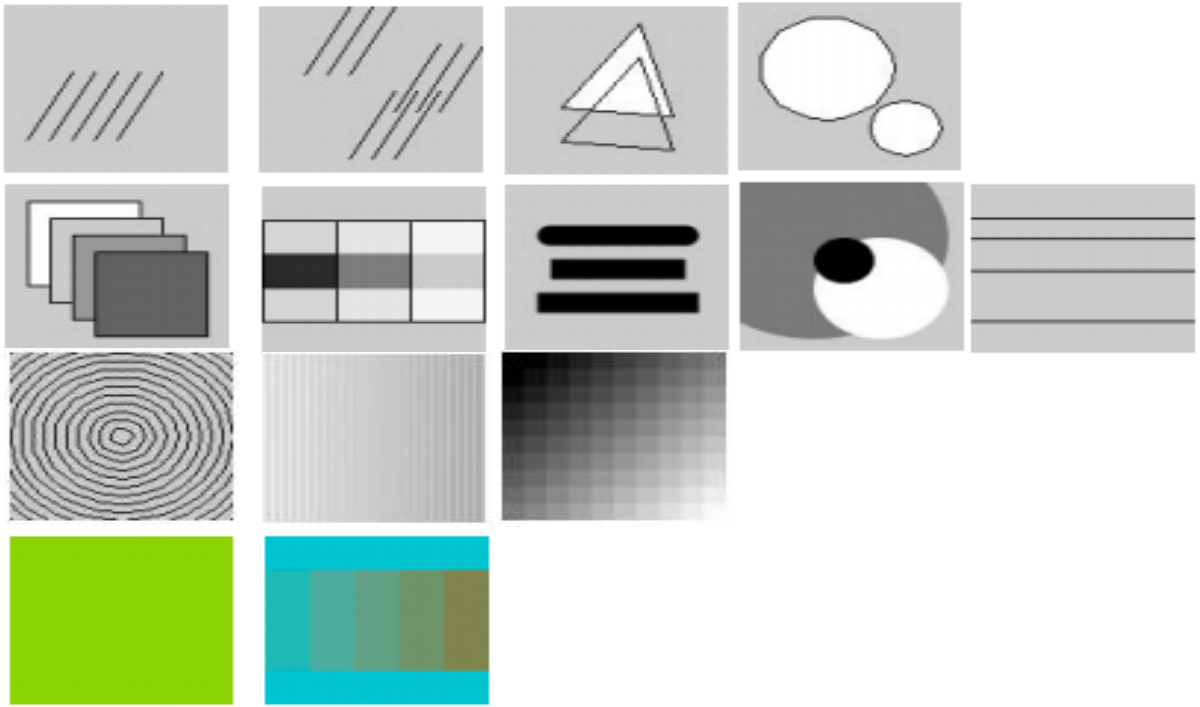
PROVA DI ESECUZIONE

- Provate a scrivere ed eseguire questo codice

```
void setup(){
    size(400,600);
    ellipse(200,100,250,100);
}
```

- Provate a cambiare i parametri per capire a cosa si riferiscono i vari numeri

SOME EXAMPLES



Processing: a programming handbook for visual designers and artists
Casey Reas, Ben Fry
MIT Press, 2007

INFORMATICA: INTRODUZIONE

COSA È L'INFORMATICA?

- Molte definizioni
  - Scienza della conservazione, dell'elaborazione e della rappresentazione automatica dell'informazione
  - Scienza dei calcolatori elettronici
  - Scienza dell'Informazione
  - Studio sistematico degli algoritmi che descrivono e trasformano l'informazione: teoria, analisi, progetto, efficienza, realizzazione e applicazione (ACM)

INFORMAZIONE

- Tutto ciò che può essere rappresentato all'interno di un computer
  - Numeri, caratteri, immagini, suoni
  - Comandi e sequenze di comandi che il calcolatore esegue per trasformare l'informazione
- Il computer (elaboratore elettronico) è lo strumento per rappresentare ed elaborare l'informazione

CALCOLATORE ELETTRONICO

Una macchina che:

- Ha un meccanismo di input per acquisire le richieste
- Ha un dispositivo per memorizzare le informazioni
- Ha la capacità di elaborare i dati
- Ha un dispositivo per comunicare al risposta

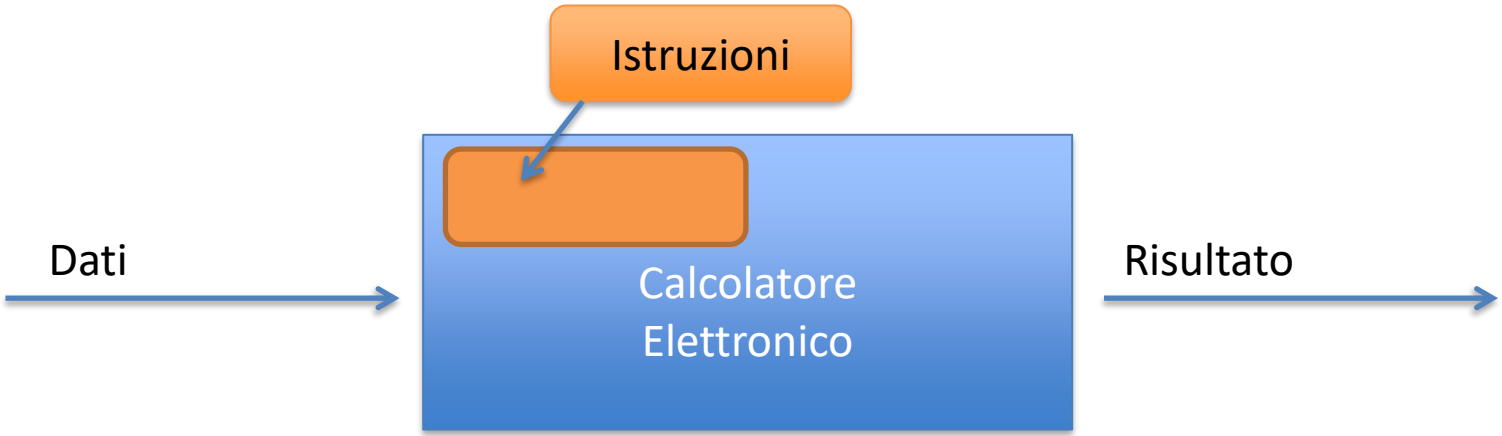
ESEMPIO: LA CALCOLATRICE

- Acquisizione delle richieste:
  - 10 tasti per le cifre
  - 4 tasti per le operazioni (funzioni) da calcolare
  - Una memoria interna per i risultati parziali
  - Un display per mostrare il risultato
- Elaborazione: un insieme di funzioni per calcolare la somma, sottrazione, divisione e moltiplicazione

- Ogni calcolatrice è limitata ad un numero predefinito di funzioni.

CALCOLATORE ELETTRONICO UNIVERSALE

- Il calcolatore può essere programmato per eseguire un insieme di azioni sui dati per risolvere un dato problema (o funzione)



FUNZIONI CALCOLABILI

- Di tutte le funzioni possibili, solo un sottoinsieme sono calcolabili, ovvero esiste un algoritmo che è in grado di fornire un risultato per i dati in input
- L'informatica si occupa di trovare risposte a domande del tipo:
  - Quante e quali sono le funzioni calcolabili?
  - Quale sono le funzioni che una data macchina può calcolare?
  - Esiste una macchina che calcoli tutte le (infinite) funzioni calcolabili?

ALGORITMO

- E’ un insieme ordinato di azioni che risolve un dato problema (funzione) P
- L’esecuzione dell’algoritmo è affidata ad un esecutore (non necessariamente un calcolatore) in grado di decifrare le azioni nella sequenza

- Nella vita quotidiana tutti eseguiamo algoritmi:
  - Montare un mobile componibile
  - Cambiare la cartuccia della stampante
  - Prendere un caffè alla macchina a gettoni

ESERCIZIO

- Pensate ad un algoritmo nella vita quotidiana

ESERCIZIO (2)

- Prevediamo tutti i passi necessari

IL MIO ALGORITMO: LA FRITTATA DI CIPOLLE

- Preparare gli ingredienti necessari: 2 cipolle rosse, 4 uova, una padella di 24 cm, un piatto fondo, un coltello affilato, una forchetta, sale, pepe e olio
- Tagliare la cipolla a fettine sottili
- Far scaldare l'olio nella padella
- Mettere la cipolla nella padella e far appassire
- Nel frattempo, sbattere le uova
- Mettere le uova nella padella
- Girare la frittata a metà cottura

IL MIO ALGORITMO: LA FRITTATA DI CIPOLLE

- Preparare gli ingredienti necessari: 2 cipolle rosse, 4 uova, una padella di 24 cm, un piatto fondo, un coltello affilato, una forchetta, sale, pepe e olio
- Tagliare la cipolla a fettine sottili
- Far scaldare l'olio nella padella
- Mettere la cipolla nella padella e far appassire
- Nel frattempo, sbattere le uova
- Mettere le uova nella padella
- Girare la frittata a metà cottura

Attrezzatura standard in una cucina

IL MIO ALGORITMO: LA FRITTATA DI CIPOLLE

- Preparare gli ingredienti necessari: 2 cipolle rosse, 4 uova, una padella di 24 cm, un piatto fondo, un coltello affilato, una forchetta, sale, pepe e olio
- Tagliare la cipolla a fettine sottili
- Far scaldare l'olio nella padella
- Mettere la cipolla nella padella e far appassire
- Nel frattempo, sbattere le uova
- Mettere le uova nella padella
- Girare la frittata a metà cottura

Dobbiamo specificare che dobbiamo pulire la cipolla? E come?
Quanto olio?

IL MIO ALGORITMO: LA FRITTATA DI CIPOLLE

- Preparare gli ingredienti necessari: 2 cipolle rosse, 4 uova, una padella di 24 cm, un piatto fondo, un coltello affilato, una forchetta, sale, pepe e olio
- Tagliare la cipolla a fettine sottili
- Far scaldare l'olio nella padella
- Mettere la cipolla nella padella e far appassire
- Nel frattempo, sbattere le uova
- Mettere le uova nella padella Ci siamo dimenticati di salarle!
- Girare la frittata a metà cottura

ALGORITMO NEL CONTESTO INFORMATICO

- A volte è complicato dividere il problema in passi elementari, che siano eseguibili da una macchina
  - Es. 1
    - Trovare il minimo tra i seguenti numeri
    - 3, 12, 8, 2, 9
  - Es. 2
    - Eseguire la somma in colonna tra due numeri
    - 55 e 49

MACCHINA ASTRATTA E ISTRUZIONI

- Un generico esecutore è associato ad un insieme finito di funzioni primitive
- Queste istruzioni sono quelle che possono essere interpretate ed eseguite

PROPRIETÀ DI UN ALGORITMO

- Finitezza. L'insieme di istruzioni devono essere finite per ogni dato in ingresso. La computazione deve terminare
- Eseguibilità. L'esecutore deve essere in grado di eseguire tutte le azioni in tempo finito e con le risorse a disposizione
- Non ambiguità. Ogni azione deve essere interpretata in modo univoco

Se almeno una di queste proprietà non è soddisfatta, la sequenza non è un algoritmo.

ALTRE PROPRIETÀ DI UN ALGORITMO

- Generalità. Corretto funzionamento dell'algoritmo anche con piccole variazioni dell'input del problema
- Efficienza. Il numero di istruzioni da eseguire deve essere il più piccolo possibile
- Determinismo. Per ogni insieme di dati in input, il risultato è corente con diverse esecuzioni dell'algoritmo

ALGORITMI E PROGRAMMAZIONE

- Algoritmo: metodo risolutivo
- Linguaggio di Programmazione: linguaggio comprensibile al calcolatore
- Programma: sequenze di istruzioni del linguaggio che descrivono un algoritmo
- Il nostro obiettivo è di ideare un algoritmo per risolvere un problema, scegliere il linguaggio di programmazione e scrivere un programma da poter far eseguire ad un calcolatore che risolva il problema

LE FASI DELLA PROGRAMMAZIONE

- Ad un primo livello di astrazione la programmazione può essere suddivisa in 4 macro-fasi:
  - Specifica: definizione del problema
    - *quale funzione si vuole calcolare e quali sono i dati di interesse?*
    - *es. dati due numeri calcolarne il maggiore*
  - Individuazione di un algoritmo (metodo risolutivo)
  - Codifica dell'algoritmo in un linguaggio di programmazione
  - Esecuzione e messa a punto (testing)

SPECIFICA

- Comprendere e definire (specificare) il problema che si vuole risolvere
- Descrizione dello stato iniziale del problema (dati iniziali, *input*) e dello stato finale atteso (risultati, *output*)
- La specifica può essere fatta in maniera più o meno rigorosa

ESEMPI DI SPECIFICA INFORMALE

- Es. 1: dati due numeri, trovare il maggiore
- Es. 2: dato un elenco telefonico e un nome, trovare il numero di telefono corrispondente
- Es. 3: data la struttura di una rete stradale e le informazioni sui flussi dei veicoli, determinare il percorso più rapido tra due posizioni A e B
- Note:
  - La descrizione non fornisce un metodo risolutivo (es. 3)
  - La descrizione può essere ambigua o imprecisa (es. 2)

ALGORITMI

- Dopo la specifica occorre individuare un algoritmo (metodo risolutivo) che permetta di ottenere i risultati attesi.
- In generale, in questa fase occorre:
  - Individuare una soluzione
  - Dimostrare che la soluzione è corretta
  - Tra più soluzioni, scegliere quella ottimale

CODIFICA

- Rappresentare l'algoritmo e l'informazione di interesse in un linguaggio di programmazione
- Il risultato è un programma eseguibile dal calcolatore
- Occorre rappresentare
  - *l'algoritmo ⇒ programma*
  - *le informazioni iniziali ⇒ dati in ingresso (input)*
  - *le informazioni finali ⇒ dati in uscita (output)*
  - *le informazioni usate dall'algoritmo ⇒ dati ausiliari*

ESECUZIONE

- Un programma può essere compilato oppure interpretato
- Compilazione
  - *il programma viene tradotto (compilato) in linguaggio macchina*
  - *il nuovo programma in linguaggio macchina viene eseguito dal calcolatore*
- Interpretazione
  - *il programma viene eseguito da un emulatore (interprete) dei comandi del linguaggio*
  - *dato il programma, è l'emulatore che ne calcola l'output*
- La correzione degli errori (in ogni caso) può richiedere la revisione di una o piú fasi (specifica, definizione dell'algoritmo, codifica)

SOMMARIO

- Informatica
- Calcolatore Elettronico
- Funzioni calcolabili
- Algoritmo

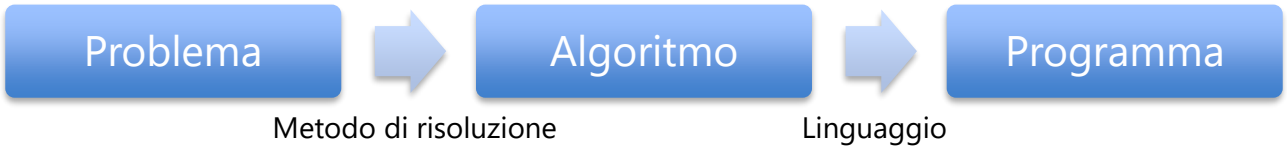
LNGUAGGI DI PROGRAMMAZIONE

ALGORITMI E LINGUAGGI DI PROGRAMMAZIONE

- Nel caso di un elaboratore elettronico è necessario:
  - Conoscere l'insieme di istruzioni che è in grado di interpretare
  - Conoscere i tipi di dati che può rappresentare
- Questi due aspetti sono centrali nella definizione di un Linguaggio di Programmazione

ALGORITMI E PROGRAMMI

- Dato un problema P, la sua soluzione si identifica secondo la seguente procedura
  - Individuare un metodo risolutivo
  - Trasformare il metodo in un insieme ordinato di azioni: algoritmo
  - Rappresentare dati e azioni attraverso un formalismo comprensibile dal calcolatore (il linguaggio di programmazione): programma
- Una volta che il programma è stato creato, potrà essere utilizzato per risolvere ogni istanza del problema P



SPECIFICA DI UN ALGORITMO: PRODOTTO DI N NUMERI

- Problema: calcolare il prodotto di N numeri
- Specifica del problema:
  - Input: N numeri $\{x_1, x_2, \ldots, x_N\}$
  - Output: un numero P tale che $P = x_1 * x_2 * \ldots * x_N$
- Specifica dell'algoritmo
  - P = 1
  - i = 1
  - finché (i<=N)
    - $P = P * x_i$
    - i = i + 1

QUALE LINGUAGGIO DI PROGRAMMAZIONE?

- I linguaggi di programmazione sono tutti equivalenti
- Dati due linguaggi L1 e L2 e due programmi che risolvono la stessa funzione f nei due linguaggi $D_{L1}(f)$ e $D_{L2}(f)$
- Allora esiste una funzione calcolabile che traduce D_{L1} in D_{L2} e viceversa

IL LINGUAGGIO MACCHINA

- Il linguaggio direttamente eseguibile da un calcolatore si chiama linguaggio macchina
- E' un linguaggio poco comprensibile per gli umani
- Le operazioni disponibili sono molto semplici
- Sono specificate in notazione binaria: sequenze di 1 e 0

- Lo codifica di una funzione complessa richiede la scrittura di un lungo programma, a volte incomprensibile
- In caso di errori o malfunzionamenti, è difficile trovare gli eventuali errori

LINGUAGGIO MACCHINA: PRODOTTO DI DUE NUMERI

Assembler	Linguaggio Macchina
READ X	0100 1000
READ Y	0100 1001
LOADA X	0000 1000
LOADB Y	0000 1001
MUL	1000
STOREA X	0010 1000
WRITE X	0101 1000
HALT	1101 0000
X INT	0000 0000
Y INT	0000 0000

LINGUAGGI DI ALTO LIVELLO

- Sono linguaggi che permettono una leggibilità e una comprensione più immediata degli algoritmi che codificano
- Sono compatti, comprensibili, modificabile e mantenibili
- I programmi di un linguaggio di alto livello devono essere tradotti in linguaggio macchina per essere eseguiti dal calcolatore
- La traduzione avviene secondo due modalità principali
  - Compilazione. Il compilatore controlla che tutte le istruzioni del programma siano corrette e alla fine genera il programma comprensibile dall'elaboratore
  - Interpretazione. L'interprete controlla una istruzione alla volta e la esegue direttamente. Appena incontra un errore, termina l'esecuzione

FASI DI SVILUPPO DI UN PROGRAMMA

- Scrivere il testo del programma e memorizzarlo su supporti permanenti (di solito file di testo sul disco del computer)
- In caso di linguaggio interpretato:
  - Usare l'interprete per eseguire il programma

- Nel nostro corso useremo Javascript e Java per Processing, due linguaggi interpretati

SOMMA DEI PRIMI 10 NUMERI

```
00110001 00000000 00000000
00110001 00000001 00000001
00110011 00000001 00000010
01010001 00001011 00000010
00100010 00000010 00001000
01000011 00000001 00000000
01000001 00000001 00000001
00010000 00000010 00000000
01100010 00000000 00000000
```

SOMMA DEI PRIMI 10 NUMERI

1. Store the number 0 in memory location 0.
2. Store the number 1 in memory location 1.
3. Store the value of memory location 1 in memory location 2.
4. Subtract the number 11 from the value in memory location 2.
5. If the value in memory location 2 is the number 0,
   continue with instruction 9.
6. Add the value of memory location 1 to memory location 0.
7. Add the number 1 to the value of memory location 1.
8. Continue with instruction 3.
9. Output the value of memory location 0.

SOMMA DEI PRIMI 10 NUMERI

```
 Set “total” to 0.
 Set “count” to 1.
[loop]
 Set “compare” to “count”.
 Subtract 11 from “compare”.
 If “compare” is zero, continue at [end].
 Add “count” to “total”.
 Add 1 to “count”.
 Continue at [loop].
[end]
 Output “total”.
```

Questo linguaggio primitivo permette di verificare se un valore è zero

Necessità di inserire etichette per controllare la sequenza di operazioni

SOMMA DEI PRIMI 10 NUMERI

```
var total = 0, count = 1;
while (count <= 10) {
  total += count;
  count += 1;
}
console.log(total);
// → 55
```

Non è necessario specificare i salti all'esterno del ciclo di somme

SOMMA DEI PRIMI 10 NUMERI

- console.log(sum(range(1, 10)));
- // → 55

Introduzione di nuovi operatori: range e sum

SOMMARIO

- Codifica di un algoritmo
- Linguaggi di programmazione ad alto livello
- Compilatori e Interpreti

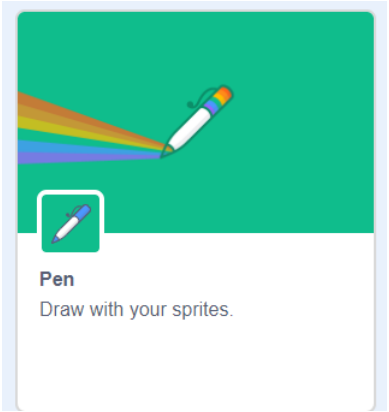
PRIME ESPERIENZE DI PROGRAMMAZIONE

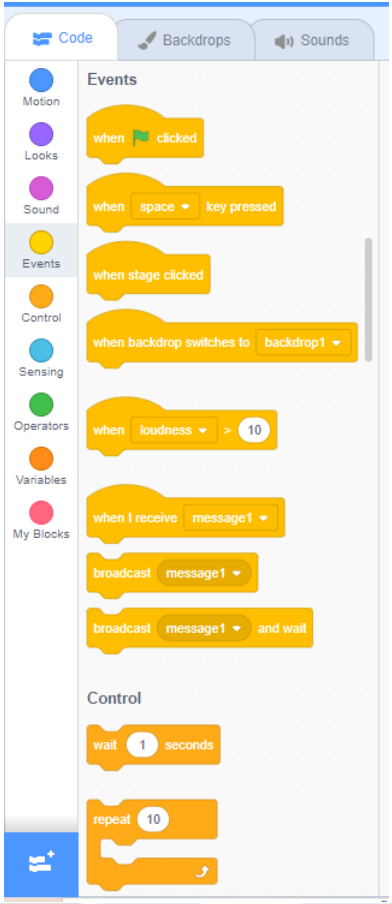
INTRODUZIONE A SCRATCH

- Scratch è un ambiente di programmazione visuale
- Invece di scrivere istruzioni su un file di testo, si usano elementi grafici per descrivere il proprio algoritmo
- Disponibile all'indirizzo:
  - https://scratch.mit.edu/projects/editor/

INTRODUZIONE A SCRATCH

- Scratch è un ambiente di programmazione visuale
- Invece di scrivere istruzioni su un file di testo, si usano elementi grafici per descrivere il proprio algoritmo
- Disponibile all'indirizzo:
  - https://scratch.mit.edu/projects/editor/
- Aggiungete l'estensione «pen»

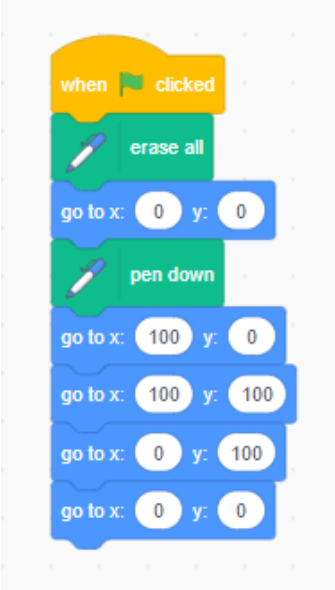




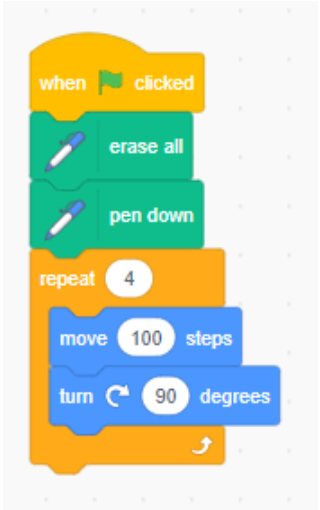
ESERCIZI

- Disegnare un quadrato di lato 100px
- Disegnare un rombo di lato 100 px
- Disegnare un esagono di lato 100px
- Disegnare un poligono regolare
- Disegnare un cerchio
- Disegnare un cerchio dato il raggio e il centro

DUE POSSIBILI MODI PER DISEGNARE UN QUADRATO



- Fisso un punto di partenza
- Mi sposto di 100px sulla destra
- Mi sposto in alto
- Mi sposto a sinistra
- Mi sposto in giù fino a tornare al punto di partenza



- Mi muovo di 100 passi
- Mi giro di 90 gradi
- Ripeto altre 3 volte

INTRODUZIONE ALLA LOGICA CLASSICA

PROPOSIZIONI LOGICHE

- Si dice proposizione logica una frase alla quale sia possibile attribuire un valore di verità secondo un criterio oggettivo

- Stabilire il valore di verità (o valore logico) di una proposizione, significa stabilire se questa è vera oppure falsa:
  - assegneremo il valore V/T a un'affermazione vera,
  - assegneremo il valore F a un'affermazione falsa.
- Esempi
  - Sono proposizioni logiche:
    - Il leone è un pesce → F
    - Il ferro è un metallo → V
    - il numero 2 è un numero pari → V
  - Non sono proposizioni logiche:
    - Smettila di piangere! (Frase esclamativa, non ha valore di verità)
    - Marco è andato al cinema? (Frase interrogativa, non ha valore di verità)
    - Questo libro è interessante (Frase il cui valore di verità è soggettivo)

I TRE PRINCIPI DELLA LOGICA

- La logica classica si basa su 3 principi fondamentali:
- Principio di identità: ogni oggetto del pensiero logico è uguale solamente a sé stesso
- Principio di non contraddizione: una proposizione logica non può essere contemporaneamente vera e falsa
- Principio del terzo escluso: una proposizione logica è sicuramente vera o falsa, non esiste una terza opzione

ALTRI TIPI DI LOGICA

- La logica a due valori (vero e falso) non è l'unica possibile
- Ad esempio, esistono:
  - la logica a tre valori: true, false, null
  - la logica fuzzy, in cui ogni proposizione ha un grado di verità

RAPPRESENTAZIONE LOGICA

- È conveniente rappresentare ogni proposizione (semplice) con un simbolo
  - p = Oggi vado all'Università
  - q = Oggi vado al mare
  - s = Oggi c'è il sole
- Possiamo così ragionare in modo più astratto

PROPOSIZIONI SEMPLICI O COMPOSTE

- Si dicono proposizioni semplici o atomiche le proposizioni costituite da un soggetto e un predicato oppure da un soggetto e un oggetto collegato da un predicato.

- Nella realtà si usano proposizioni più complesse che si ottengono collegando le proposizioni semplici mediante particelle linguistiche come non, e, o, o...o, se...allora, se e solo se. Ognuna di queste particelle si dice connettivo logico.

- Si dicono proposizioni composte o molecolari le proposizione ottenute collegando due o più proposizioni semplici mediante connettivi logici.

CONNETTIVI LOGICI

- I connettivi logici rappresentano le particelle che permettono di indicare alcune condizioni che legano proposizioni semplici
  - non, e, o, o...o, se...allora, se e solo se
- Hanno degli specifici simboli in logica
  - ¬, ∧, ∨, ⊕, →, ↔
- Permettono di creare proposizioni logiche composte
- Sono universali e indipendenti dal linguaggio
- Hanno un significato preciso

Le tabelle di verità

- Le tabelle di verità sono tabelle usate nella logica per determinare se, attribuiti i valori di verità alle proposizioni che la compongono, una determinata proposizione è vera o falsa
- La tabella di verità quindi si applica a qualsiasi operatore logico
- Le condizioni di verità o falsità di qualunque enunciato che si ottiene applicando quell'operatore è determinato interamente ed esclusivamente da quelle degli enunciati più semplici a cui si applica
- Per costruirle dobbiamo prevedere tutte le combinazioni dei valori vero/falso di tutti gli enunciati semplici

Negazione

- La negazione di una proposizione *p* è la proposizione "**non** *p*", risulta vera se p è falsa e falsa se p è vera.
- La negazione si indica con il simbolo ¬ anteposto alla proposizione da negare.
- Esempio. Se *p*: "Oggi vado all'università", allora ¬*p*: "Oggi non vado all'università".

p	¬p
T	F
F	T

CONGIUNZIONE

- La congiunzione di due proposizioni *p* e *q* è la proposizione "*p* **e** *q*", risulta vera solo se le due proposizioni sono entrambe vere. Basta che una delle due sia falsa perché sia falsa
- La congiunzione si indica con il simbolo ∧.
- Esempio. Se *p*: "Oggi vado all'università" e *q*: "Oggi vado al mare" allora *p* ∧ *q*: "Oggi vado all'università e al mare".

p	**q**	**p ∧ q**
T	T	T
T	F	F
F	T	F
F	F	F

DISGIUNZIONE

- La disgiunzione di due proposizioni *p* e *q* è la proposizione "*p* **o** *q*", essa è falsa solo se le due proposizioni sono entrambe false. Basta che una delle due sia vera perché sia vera.
- La disgiunzione si indica con il simbolo ∨.
- Esempio. Se *p*: "Oggi vado all'università" e *q*: "Oggi vado al mare" allora *p* ∨ *q*: "Oggi vado all'università o al mare".

p	**q**	**p ∨ q**
T	T	T
T	F	T
F	T	T
F	F	F

DISGIUNZIONE ESCLUSIVA

- La disgiunzione esclusiva di due proposizioni *p* e *q* è la proposizione "**o** *p* **o** *q*", essa è vera solo se una proposizione è vera e l'altra è falsa.
- La disgiunzione esclusiva si indica con il simbolo $\oplus$.
- Esempio. Se *p*: "Oggi vado all'università" e *q*: "Oggi vado al mare" allora $p \oplus q$: "Oggi o vado all'università o vado al mare"

p	**q**	**p $\oplus$ q**
T	T	F
T	F	T
F	T	T
F	F	F

IMPLICAZIONE

- L'implicazione di due proposizioni *p* e *q* è la proposizione "**se** *p* **allora** *q*", essa risulta falsa solo se *p* è vera e *q* è falsa.
- L'implicazione si indica con il simbolo → o ⇒.
- Esempio. Se *p*: "Oggi c'è il sole" e *q*: "Oggi vado al mare" allora *p* → *q*: "Se oggi c'è il sole allora vado al mare".

p	**q**	**p → q**
T	T	T
T	F	F
F	T	T
F	F	T

LA DOPPIA IMPLICAZIONE

- La doppia implicazione di due proposizioni p e q è la proposizione "p **se e solo se** q", è vera solo se p e q sono entrambe vere o entrambe false.
- La doppia implicazione si indica con il simbolo $\leftrightarrow$ o $\Leftrightarrow$.
- Esempio. Se p: "Oggi vado al mare" e q: "Oggi c'è il sole" allora $p \leftrightarrow q$: "Oggi vado al mare se e solo se c'è il sole"

p	q	p ↔ q
T	T	T
T	F	F
F	T	F
F	F	T

Infatti:

p	q	p → q	q → p	(p → q) ∧ (q → p)
T	T	T	T	T
T	F	F	T	F
F	T	T	F	F
F	F	T	T	T

USO DI TABELLE DI VERITÀ

- Le tabelle di verità consentono di verificare (in modo formale) quando una certa proposizione (complessa quanto si vuole) è vera o falsa
- $(\neg p \vee (p \wedge q)) \oplus \neg(p \rightarrow q)$

		1	2	3	4	5	
p	**q**	**¬p**	**p ∧ q**	**1 ∨ 2**	**p→q**	**¬(p→q)**	**3 ⊕ 5**
T	T	F	T	T	T	F	T
T	F	F	F	F	F	T	T
F	T	T	F	T	T	F	T
F	F	T	F	T	T	F	T

EQUIVALENZA DI ESPRESSIONI LOGICHE

- Due proposizioni sono equivalenti se e solo se hanno gli stessi valori di verità, ovvero la loro tavola di verità coincide.

- Alcune equivalenze sono:
  - $\neg\neg p = p$
  - $p \wedge (q \vee r) = (p \wedge q) \vee (p \wedge r)$
  - $p \vee (q \wedge r) = (p \vee q) \wedge (p \vee r)$
  - $(p \Rightarrow q) = (\neg q \Rightarrow \neg p)$ (proposizione contronominale)
  - $\neg(p \wedge q) = \neg p \vee \neg q$
  - $\neg(p \vee q) = \neg p \wedge \neg q$

  } Leggi di De Morgan

ESERCIZIO

- Dimostrare le seguenti equivalenze con le tabelle di verità
  - $\neg\neg p = p$
  - $p \wedge (q \vee r) = (p \wedge q) \vee (p \wedge r)$
  - $p \vee (q \wedge r) = (p \vee q) \wedge (p \vee r)$
  - $(p \Rightarrow q) = (\neg q \Rightarrow \neg p)$
  - $\neg(p \wedge q) = \neg p \vee \neg q$
  - $\neg(p \vee q) = \neg p \wedge \neg q$

SOMMARIO

- Proposizioni logiche
- Connettivi logici
- Tabelle di verità